```
> restart;
  with(Statistics):
> Digits:=40;
  interface(rtablesize=25);
```

$$Digits := 40$$

$$10 \tag{1}$$

3-connected->2-connected

Definitions

Expressions of generating functions of bicolored binary trees

We obtain expressions of the generating functions of bicolored binary trees and their partial derivatives. As they will be evaluated at (x,D) in the systems given after, we replace y by D. These expressions will be placed in the systems verified by networks, pointed networks, and bi-pointed networks. Here u and v stand respectively for the generating functions of black-rooted and white-rooted bicolored binary trees.

```
> system_u_v:={u=x*D*(1+v)^2,v=D*(1+u)^2};
```

$$system\_u\_v := \left\{u = x\,\mathrm{D}\,(1+v)^2,\, v = \mathrm{D}\,(1+u)^2\right\} \tag{1.1.1.1}$$

```
> Equation_dxu_dxv:=subs({diff(u(x,D),x)=dxu,diff(v(x,D),x)=
  dxv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,D),v=v(x,D),
  system_u_v),x));
  solution_dxu_dxv_u_v:=solve(Equation_dxu_dxv,{dxu,dxv});
  Equation_dyu_dyv:=subs({diff(u(x,D),D)=dyu,diff(v(x,D),D)=
  dyv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,D),v=v(x,D),
  system_u_v),D));
  solution_dyu_dyv_u_v:=solve(Equation_dyu_dyv,{dyu,dyv});
  system_derivate_binary_tree:={
     dxu=subs(solution_dxu_dxv_u_v,dxu),
     dxv=subs(solution_dxu_dxv_u_v,dxv),
     dyu=subs(solution_dyu_dyv_u_v,dyu),
     dyv=subs(solution_dyu_dyv_u_v,dyv)
  };
```

$$Equation\_dxu\_dxv := \left\{dxu = \mathrm{D}\,(1+v)^2 + 2\,x\,\mathrm{D}\,(1+v)\,dxv,\, dxv = 2\,\mathrm{D}\,(1+u)\,dxu\right\}$$

$$solution\_dxu\_dxv\_u\_v := \left\{dxu = -\frac{\mathrm{D}\,(1+2\,v+v^2)}{-1+4\,x\,\mathrm{D}^2+4\,x\,\mathrm{D}^2\,u+4\,x\,\mathrm{D}^2\,v+4\,x\,\mathrm{D}^2\,v\,u},\, dxv =\right.$$

$$-\frac{2\,\mathrm{D}^2\,(1+u)\,\left(1+2\,v+v^2\right)}{-1+4\,x\,\mathrm{D}^2+4\,x\,\mathrm{D}^2\,u+4\,x\,\mathrm{D}^2\,v+4\,x\,\mathrm{D}^2\,v\,u}\Bigg\}$$

$$Equation\_dyu\_dyv := \left\{dyu = x\,(1+v)^2 + 2\,x\,\mathrm{D}\,(1+v)\,dyv,\ dyv = (1+u)^2 + 2\,\mathrm{D}\,(1+u)\,dyu\right\}$$

$$solution\_dyu\_dyv\_u\_v := \Bigg\{dyu = -\frac{x\left(2\,\mathrm{D}\,v\,u^2+1+2\,v+v^2+2\,\mathrm{D}\,u^2+2\,\mathrm{D}+4\,\mathrm{D}\,u+4\,\mathrm{D}\,v\,u+2\,\mathrm{D}\,v\right)}{-1+4\,x\,\mathrm{D}^2+4\,x\,\mathrm{D}^2\,u+4\,x\,\mathrm{D}^2\,v+4\,x\,\mathrm{D}^2\,v\,u},$$

$$dyv = -\left(1+2\,u+2\,x\,\mathrm{D}\,u\,v^2+u^2+2\,x\,\mathrm{D}+2\,x\,\mathrm{D}\,u+2\,x\,\mathrm{D}\,v^2+4\,x\,\mathrm{D}\,v+4\,x\,\mathrm{D}\,v\,u\right)\Big/\left(-1+4\,x\,\mathrm{D}^2+4\,x\,\mathrm{D}^2\,u+4\,x\,\mathrm{D}^2\,v+4\,x\,\mathrm{D}^2\,v\,u\right)\Bigg\}$$

$$system\_derivate\_binary\_tree := \Bigg\{dxu = -\frac{\mathrm{D}\left(1+2\,v+v^2\right)}{-1+4\,x\,\mathrm{D}^2+4\,x\,\mathrm{D}^2\,u+4\,x\,\mathrm{D}^2\,v+4\,x\,\mathrm{D}^2\,v\,u},\ dxv = -\frac{2\,\mathrm{D}^2\,(1+u)\left(1+2\,v+v^2\right)}{-1+4\,x\,\mathrm{D}^2+4\,x\,\mathrm{D}^2\,u+4\,x\,\mathrm{D}^2\,v+4\,x\,\mathrm{D}^2\,v\,u},$$ **(1.1.1.2)**

$$dyu = -\frac{x\left(2\,\mathrm{D}\,v\,u^2+1+2\,v+v^2+2\,\mathrm{D}\,u^2+2\,\mathrm{D}+4\,\mathrm{D}\,u+4\,\mathrm{D}\,v\,u+2\,\mathrm{D}\,v\right)}{-1+4\,x\,\mathrm{D}^2+4\,x\,\mathrm{D}^2\,u+4\,x\,\mathrm{D}^2\,v+4\,x\,\mathrm{D}^2\,v\,u},$$

$$dyv = -\left(1+2\,u+2\,x\,\mathrm{D}\,u\,v^2+u^2+2\,x\,\mathrm{D}+2\,x\,\mathrm{D}\,u+2\,x\,\mathrm{D}\,v^2+4\,x\,\mathrm{D}\,v+4\,x\,\mathrm{D}\,v\,u\right)\Big/\left(-1+4\,x\,\mathrm{D}^2+4\,x\,\mathrm{D}^2\,u+4\,x\,\mathrm{D}^2\,v+4\,x\,\mathrm{D}^2\,v\,u\right)\Bigg\}$$

```
> system_bi_derivate_binary_tree:={
    dxxu=factor(subs(system_derivate_binary_tree,subs({diff(u
  (x,D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs
  ({u=u(x,D),v=v(x,D)},subs(system_derivate_binary_tree,dxu)),
  x)))),
    dxxv=factor(subs(system_derivate_binary_tree,subs({diff(u
  (x,D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs
  ({u=u(x,D),v=v(x,D)},subs(system_derivate_binary_tree,dxv)),
  x)))),
    dxyu=factor(subs(system_derivate_binary_tree,subs({diff(u
```

```
(x,D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs
({u=u(x,D),v=v(x,D)},subs(system_derivate_binary_tree,dyu)),
x)))),
   dxyv=factor(subs(system_derivate_binary_tree,subs({diff(u
(x,D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs
({u=u(x,D),v=v(x,D)},subs(system_derivate_binary_tree,dyv)),
x)))),
   dyyu=factor(subs(system_derivate_binary_tree,subs({diff(u
(x,D),D)=dyu,diff(v(x,D),D)=dyv,u(x,D)=u,v(x,D)=v},diff(subs
({u=u(x,D),v=v(x,D)},subs(system_derivate_binary_tree,dyu)),
D)))),
   dyyv=factor(subs(system_derivate_binary_tree,subs({diff(u
(x,D),D)=dyu,diff(v(x,D),D)=dyv,u(x,D)=u,v(x,D)=v},diff(subs
({u=u(x,D),v=v(x,D)},subs(system_derivate_binary_tree,dyv)),
D))))
};
> system_tri_derivate_binary_tree:={
dxxxu=factor(subs(system_derivate_binary_tree,subs({diff(u
(x,D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs
({u=u(x,D),v=v(x,D)},subs(system_bi_derivate_binary_tree,
dxxu)),x)))),
dxxxv=factor(subs(system_derivate_binary_tree,subs({diff(u
(x,D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs
({u=u(x,D),v=v(x,D)},subs(system_bi_derivate_binary_tree,
dxxv)),x)))),
dxxyu=factor(subs(system_derivate_binary_tree,subs({diff(u
(x,D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs
({u=u(x,D),v=v(x,D)},subs(system_bi_derivate_binary_tree,
dxyu)),x)))),
dxxyv=factor(subs(system_derivate_binary_tree,subs({diff(u
(x,D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs
({u=u(x,D),v=v(x,D)},subs(system_bi_derivate_binary_tree,
dxyv)),x)))),
dxyyu=factor(subs(system_derivate_binary_tree,subs({diff(u
(x,D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs
({u=u(x,D),v=v(x,D)},subs(system_bi_derivate_binary_tree,
dyyu)),x)))),
dxyyv=factor(subs(system_derivate_binary_tree,subs({diff(u
(x,D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs
({u=u(x,D),v=v(x,D)},subs(system_bi_derivate_binary_tree,
dyyv)),x)))),
dyyyu=factor(subs(system_derivate_binary_tree,subs({diff(u
(x,D),D)=dyu,diff(v(x,D),D)=dyv,u(x,D)=u,v(x,D)=v},diff(subs
({u=u(x,D),v=v(x,D)},subs(system_bi_derivate_binary_tree,
dyyu)),D)))),
dyyyv=factor(subs(system_derivate_binary_tree,subs({diff(u
(x,D),D)=dyu,diff(v(x,D),D)=dyv,u(x,D)=u,v(x,D)=v},diff(subs
({u=u(x,D),v=v(x,D)},subs(system_bi_derivate_binary_tree,
```

```
dyyv)),D))))
}:
```

Expressions of generating functions of 3-connected networks

K is the generating function of networks such that the associated graph, obtained by adding the root edge, is 3-connected. K is equal to M/(2*x^2*y). We obtain expressions of K and its partial derivatives with respect to the generating functions of bicolored binary trees. As they will be evaluated at (x,D) in the systems given after, we replace y by D. These expressions will be placed in the systems verified by networks, pointed networks, and bi-pointed networks.

```
> system_3_connected:={K=subs(y=D,1/(2*x^2*y)*x^2*y^2*(1/(1+x*
y)+1/(1+y)-1-(1+u)^2*(1+v)^2/(1+u+v)^3))};
```

$$system\_3\_connected := \left\{K = \frac{1}{2}\,\mathrm{D}\left(\frac{1}{1+x\,\mathrm{D}} + \frac{1}{1+\mathrm{D}} - 1 - \frac{(1+u)^2\,(1+v)^2}{(1+u+v)^3}\right)\right\} \tag{1.1.2.1}$$

```
> system_derivate_3_connected:={
dxK=subs(system_derivate_binary_tree,subs({diff(u(x,D),x)=
dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,
D),v=v(x,D),subs(system_3_connected,K)),x))),
dyK=subs(system_derivate_binary_tree,subs({diff(u(x,D),D)=
dyu,diff(v(x,D),D)=dyv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,
D),v=v(x,D),subs(system_3_connected,K)),D)))
}:
> system_bi_derivate_3_connected:={
dxxK=subs(system_derivate_binary_tree,subs({diff(u(x,D),x)=
dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,
D),v=v(x,D),subs(system_derivate_3_connected,dxK)),x))),
dxyK=subs(system_derivate_binary_tree,subs({diff(u(x,D),x)=
dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,
D),v=v(x,D),subs(system_derivate_3_connected,dyK)),x))),
dyyK=subs(system_derivate_binary_tree,subs({diff(u(x,D),D)=
dyu,diff(v(x,D),D)=dyv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,
D),v=v(x,D),subs(system_derivate_3_connected,dyK)),D)))
}:
> system_tri_derivate_3_connected:={
dxxxK=subs(system_derivate_binary_tree,subs({diff(u(x,D),x)=
dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,
D),v=v(x,D),subs(system_bi_derivate_3_connected,dxxK)),x))),
dxxyK=subs(system_derivate_binary_tree,subs({diff(u(x,D),x)=
dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,
D),v=v(x,D),subs(system_bi_derivate_3_connected,dxyK)),x))),
dxyyK=subs(system_derivate_binary_tree,subs({diff(u(x,D),D)=
```

```
dyu,diff(v(x,D),D)=dyv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,
D),v=v(x,D),subs(system_bi_derivate_3_connected,dxyK)),D))),
dyyyK=subs(system_derivate_binary_tree,subs({diff(u(x,D),D)=
dyu,diff(v(x,D),D)=dyv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,
D),v=v(x,D),subs(system_bi_derivate_3_connected,dyyK)),D)))
}:
```

Expressions of generating functions of networks and their derivatives

Networks

```
> system_networks:={
D=y+S+P+H,
S=(y+P+H)*x*D,
P=y*(exp(S+H)-1)+(exp(S+H)-S-H-1),
H=K,

u=x*D*(1+v)^2,
v=D*(1+u)^2,
K=1/2*D*(1/(1+x*D)+1/(1+D)-1-(1+u)^2*(1+v)^2/(1+u+v)^3)
}:
> eval_networks:=proc(x0,y0)
global system_networks:
fsolve(subs({x=x0,y=y0},system_networks),{u,v,K,S,P,H,D});
end proc:
```

Derivate networks

```
> equation_derivate_networks:={
dD=dS+dP+dH,
dS=(dP+dH)*x*D+(y+P+H)*D+(y+P+H)*x*dD,
dP=y*(dS+dH)*exp(S+H)+(dS+dH)*(exp(S+H)-1),
dH=dxK+dD*dyK
}:
> system_derivate_networks:=solve
(equation_derivate_networks,{dD,dS,dP,dH}):
```

Bi-derivate networks

```
> equation_bi_derivate_networks:={
ddD=ddS+ddP+ddH,
ddS=(ddP+ddH)*x*D+2*(dP+dH)*D+2*(dP+dH)*x*dD+2*(y+P+H)*dD+
(y+P+H)*x*ddD,
ddP=y*(ddS+ddH)*exp(S+H)+y*(dS+dH)^2*exp(S+H)+(ddS+ddH)*
(exp(S+H)-1)+(dS+dH)^2*exp(S+H),
ddH=dxxK+dD*dxyK+ddD*dyK+dD*dxyK+dD^2*dyyK
}:
> system_bi_derivate_networks:=solve
```

```
(equation_bi_derivate_networks,{ddD,ddS,ddP,ddH}):
```

Tri-derivate networks

```
> equation_tri_derivate_networks:={
dddD=dddS+dddP+dddH,
dddS=(dddP+dddH)*x*D+(ddP+ddH)*D+(ddP+ddH)*x*dD +2*(ddP+
ddH)*D+2*(dP+dH)*dD   + 2*(ddP+ddH)*x*dD+2*(dP+dH)*dD+2*
(dP+dH)*x*ddD+2*(dP+dH)*dD+2*(y+P+H)*ddD+(dP+dH)*x*ddD+(y+
P+H)*ddD+(y+P+H)*x*dddD,
dddP=y*(dddS+dddH)*exp(S+H)+y*(ddS+ddH)*(dS+dH)*exp(S+H)
+2*y*(ddS+ddH)*(dS+dH)*exp(S+H)+y*(dS+dH)^3*exp(S+H)   +
(dddS+dddH)*(exp(S+H)-1)+(ddS+ddH)*(dS+dH)*exp(S+H)   +2*
(ddS+ddH)*(dS+dH)*exp(S+H)+(dS+dH)^3*exp(S+H),
dddH=
dxxxK+dD*dxxyK+ddD*dxyK+dD*dxxyK+dD^2*dxyyK +dddD*dyK+ddD*
dxyK+ddD*dD*dyyK   +ddD*dxyK+dD*dxxyK+dD^2*dxyyK   +2*ddD*
dD*dyyK+dD^2*dxyyK+dD^3*dyyyK
}:
```

```
> system_tri_derivate_networks:=solve
(equation_tri_derivate_networks,{dddD,dddS,dddP,dddH}):
```

Finding the singularity of x->D(x,y)

We follow the notations of Gimenez-Noy for the system giving the singularity of x->D(x,y)

```
> system_singularity_networks:={
xsi=(1+3*t)*(1-t)^3/(16*t^3),
Y=(1+2*t)/((1+3*t)*(1-t))*exp(-(t^2*(1-t)*(18+36*t+5*t^2))/
(2*(3+t)*(1+2*t)*(1+3*t)^2))-1
};
```

$$system\_singularity\_networks := \left\{ Y = \frac{(1+2\,t)\,\mathrm{e}^{-\frac{1}{2}\,\frac{t^2\,(1-t)\,(18+36\,t+5\,t^2)}{(3+t)\,(1+2\,t)\,(1+3\,t)^2}}}{(1+3\,t)\,(1-t)} - 1,\ xsi = \frac{1}{16}\,\frac{(1+3\,t)\,(1-t)^3}{t^3} \right\} \tag{1.1.4.1}$$

```
> find_singularity:=proc(Y0)
fsolve(subs(Y=Y0,system_singularity_networks),{t,xsi});
end proc;
```

$$find\_singularity := \mathbf{proc}(Y0)\ fsolve(subs(Y = Y0, system\_singularity\_networks), \{t, xsi\})\ \mathbf{end\ proc} \tag{1.1.4.2}$$

```
> find_singularity(1);
```

$$\{t = 0.62637166330645166589299785045039561167\,21, xsi = 0.038191097669411335391152564045422359553\,88\} \tag{1.1.4.3}$$

When generating only networks, calculation of the choose-vectors for the Boltzmann samplers of binary trees and networks

Choose of the size of the networks

The real y0 has to be chosen to give a desired balance between the number of vertices and number of edges. See the curve of mu(t) given by Bender, Gao and Wormald

```
> y0:=1;
```

$$y0 := 1 \tag{2.1.1}$$

Then we choose a size N around which we want the boltzmann sampler for bi-pointed networks to have a good chance of picking up networks of this size

```
> N:=10000;
```

$$N := 10000 \tag{2.1.2}$$

We calculate the real x0 for which the Boltzmann sampler of bi-pointed networks has good chances of picking up networks of this size. This value x0 verifies x=R(1-1/2N) where R is the singularity of x->D(x,y0)

```
> R:=subs(find_singularity(y0),xsi);
  x0:=evalf(R*(1-1/(2*N)));
```

$$R := 0.038191097669411335391152564045422359553 88$$

$$x0 := 0.038189188114527864824383006417220088435 90 \tag{2.1.3}$$

Calculating the choose-vectors of bicolored binary trees

```
> solution_networks:=eval_networks(x0, y0);
```

$$solution\_networks := \{D = 1.0941677775676487822661125631858619495 02, H = 0.0021226463933494267222707663829796725181745, K = 0.0021226463933494267222707663829796725181745, P = 0.048158725886227711816506497273084593477 28, S = 0.043886405288071643726898402082980630843 28, u = 0.525592882166856422983538528232029982855 7, v = 2.5466028956326222931861004821395501196725\} \tag{2.2.1}$$

```
> solution_derivate_3_connected:=subs(x=x0,solution_networks,
  system_derivate_3_connected);
  solution_bi_derivate_3_connected:=subs(x=x0,solution_networks,
  system_bi_derivate_3_connected);
  solution_tri_derivate_3_connected:=subs(x=x0,
  solution_networks,system_tri_derivate_3_connected);
```

$$solution\_derivate\_3\_connected := \{dxK = 0.25004179403267751160592997841878691460 52, dyK = 0.021475899686895129108777243991068343971 86\}$$

$solution\_bi\_derivate\_3\_connected := \{dxxK$
$= 857.80777062361948401864980558900463074 5, dxyK$
$= 73.605899434045996327694505390044567173 70, dyyK$
$= 6.2498119074041484140394666282926009902 30\}$

$solution\_tri\_derivate\_3\_connected := \{dxxxK$ **(2.2.2)**
$= 1.7501317823092695618911570212272415449 92\ 10^8, dxxyK$
$= 1.4905561111028766284160580115591503607 73\ 10^7, dxyyK$
$= 1.2692596512848176376776252647663565752 08\ 10^6, dyyyK$
$= 1.0806285898030604863856871756540775088 16\ 10^5\}$

```
> solution_derivate_binary_tree:=subs(x=x0,solution_networks,
  system_derivate_binary_tree);
  solution_bi_derivate_binary_tree:=subs(x=x0,solution_networks,
  system_bi_derivate_binary_tree);
  solution_tri_derivate_binary_tree:=subs(x=x0,
  solution_networks,system_tri_derivate_binary_tree);
```

$solution\_derivate\_binary\_tree := \{dxu$
$= 1311.7967072607223427435889737778211050 53, dxv$
$= 4379.4453058101334633588470384652603977 24, dyu$
$= 111.53588115165386619932263506867120471 01, dyv$
$= 374.69099305294689007207385658230680165 91\}$

$solution\_bi\_derivate\_binary\_tree := \{dxxu$
$= 2.6563637466369098959049743055718952787 28\ 10^8, dxxv$
$= 8.9059517753505368667885582820607788441 62\ 10^8, dxyu$
$= 2.2620609614841676590554717522205877157 89\ 10^7, dxyv$
$= 7.5843295269335291322208882039123820539 42\ 10^7, dyyu$
$= 1.9259494377174596738985170654245565059 17\ 10^6, dyyv$
$= 6.4577038947833369561252981141249762163 99\ 10^6\}$

$solution\_tri\_derivate\_binary\_tree := \{dxxxu$ **(2.2.3)**
$= 1.5981896413729386393614320843914843858 38\ 10^{14}, dxxxv$
$= 5.3584472186262375508977853895803218933 89\ 10^{14}, dxxyu$
$= 1.3609153415060281384325156669136138673 40\ 10^{13}, dxxyv$
$= 4.5629804848509155769264963855522274730 53\ 10^{13}, dxyyu$

$= 1.1587993609362340766251926413799122865 12\ 10^{12}$, *dxyyv*

$= 3.8853720030395430327866473481240544567 52\ 10^{12}$, *dyyyu*

$= 9.8664227890942652969228668553830805457 51\ 10^{10}$, *dyyyv*

$= 3.3081937647827713378517787560600519123 49\ 10^{11}$}

```
> solution_derivate_networks:=evalf(subs(x=x0,y=y0,
  solution_networks,solution_derivate_3_connected,
  system_derivate_networks));
  solution_bi_derivate_networks := evalf(subs(x=x0,y=y0,
  solution_networks,solution_derivate_3_connected,
  solution_derivate_networks, solution_bi_derivate_3_connected,
  system_bi_derivate_networks));
  solution_tri_derivate_networks := evalf(subs(x=x0,y=y0,
  solution_networks,solution_derivate_3_connected,
  solution_derivate_networks, solution_bi_derivate_3_connected,
  solution_tri_derivate_3_connected,
  solution_bi_derivate_networks, system_tri_derivate_networks));
```

solution_derivate_networks := {*dD* = 3.5851154226657975699439164722436940014 27,

dH = 0.32703537321578881209438248500310139327 78, *dP*

= 1.8731630848116324544698017513296576904 29, *dS*

= 1.3849169646383763033797322359109349177 20}

solution_bi_derivate_networks := {*ddD*

= 3924.6079652238474536961260237202062749 90, *ddH*

= 1550.1927157017199188430406340887043597 27, *ddP*

= 2053.4730635848799327684462962535430421 81, *ddS*

= 320.94218593724760208463909337795887308 25}

solution_tri_derivate_networks := {*dddD* **(2.2.4)**

$= 1.0344001747380388365227331630728241106 00\ 10^{9}$, *dddH*

$= 4.1259383918696967977174058788751817320 78\ 10^{8}$, *dddP*

$= 5.4046647438835131541146078523554226647 19\ 10^{8}$, *dddS*

$= 8.1339861162717841339531789949763670919 64\ 10^{7}$}

```
> ch_1_or_u_Bernoulli:=subs(solution_networks,[1/(1+u),u/(1+u)])
  ;
  ch_1_or_u:=CumulativeSum(ch_1_or_u_Bernoulli);
```

ch_1_or_u_Bernoulli := [0.6554828694400191053580522747732445925295,

0.3445171305599808946419477252267554074703]

(2.2.5)

$ch\_1\_or\_u := [0.65548286944001910535805227477324459252 95,$ (2.2.5)

$0.99999999999999999999999999999999999998]$

```
> ch_1_or_v_Bernoulli:=subs(solution_networks,[1/(1+v),v/(1+v)])
  ;
  ch_1_or_v:=CumulativeSum(ch_1_or_v_Bernoulli);
```

$ch\_1\_or\_v\_Bernoulli := [0.28195995701447868618121612447425565025 35,$

$0.71804004298552131381878387552574434974 65]$

$ch\_1\_or\_v := [0.28195995701447868618121612447425565025 35,$ (2.2.6)

$1.0000000000000000000000000000000000000]$

```
> ch_u_or_v_Bernoulli:=subs(solution_networks,[u/(u+v),v/(u+v)])
  ;
  ch_u_or_v:=CumulativeSum(ch_u_or_v_Bernoulli);
```

$ch\_u\_or\_v\_Bernoulli := [0.17108053007719536070795135631552945935 28,$

$0.82891946992280463929204864368447054064 71]$

$ch\_u\_or\_v := [0.17108053007719536070795135631552945935 28,$ (2.2.7)

$0.99999999999999999999999999999999999999]$

Calculating the choose-vectors of pointed bicolored binary trees

Here pU and pV stand for the generating functions of black-rooted and white-rooted bicolored binary trees with a pointed black vertex

```
> ch_dxu_or_dxv_Bernoulli:=subs(solution_derivate_binary_tree,
  [dxu/(dxu+dxv),dxv/(dxu+dxv)]);
  ch_dxu_or_dxv:=CumulativeSum(ch_dxu_or_dxv_Bernoulli);
```

$ch\_dxu\_or\_dxv\_Bernoulli := [0.23049392456830503614652516445512912531 96,$

$0.76950607543169496385347483554487087468 04]$

$ch\_dxu\_or\_dxv := [0.23049392456830503614652516445512912531 96,$ (2.3.1)

$1.0000000000000000000000000000000000000]$

```
> choose_vector_dxu_Bernoulli:=subs(x=x0,solution_networks,
  solution_derivate_binary_tree, [(1+v)^2*D/dxu,x*D*dxv*(1+v)
  /dxu,x*D*dxv*(1+v)/dxu]);
  choose_vector_dxu:=CumulativeSum(choose_vector_dxu_Bernoulli);
```

$choose\_vector\_dxu\_Bernoulli := [0.01049161905385143899819365011607635612000,$

$0.49475419047307428050090317494196182194 00,$

$0.49475419047307428050090317494196182194 00]$

$choose\_vector\_dxu := [0.01049161905385143899819365011607635612000,$ (2.3.2)

$0.50524580952692571949909682505803817806 00,$

$1.0000000000000000000000000000000000000]$

```
> choose_vector_dxv_Bernoulli:=subs(solution_networks,
```

```
solution_derivate_binary_tree,[D*dxu*(1+u)/dxv,(1+u)*D*
dxu/dxv]);
choose_vector_dxv:=CumulativeSum(choose_vector_dxv_Bernoulli);
```

choose_vector_dxv_Bernoulli := [0.50000000000000000000000000000000000005, 0.50000000000000000000000000000000000005]

choose_vector_dxv := [0.50000000000000000000000000000000000005, 1.0000000000000000000000000000000000001] **(2.3.3)**

```
> ch_dyu_or_dyv_Bernoulli:=subs(solution_derivate_binary_tree,
[dyu/(dyu+dyv),dyv/(dyu+dyv)]);
ch_dyu_or_dyv:=CumulativeSum(ch_dyu_or_dyv_Bernoulli);
```

ch_dyu_or_dyv_Bernoulli := [0.2293906138654120815232837852494691211930, 0.7706093861345879184767162147505308788070]

ch_dyu_or_dyv := [0.2293906138654120815232837852494691211930, 1.0000000000000000000000000000000000000] **(2.3.4)**

```
> choose_vector_dyv_Bernoulli := subs(solution_networks,
solution_derivate_binary_tree, [(1+u)^2/dyv, dyu*D*(1+u)/dyv,
(1+u)*D*dyu/dyv]);
choose_vector_dyv:=CumulativeSum(choose_vector_dyv_Bernoulli);
```

choose_vector_dyv_Bernoulli := [0.006211608192538779629341215804977867172935, 0.4968941959037306101853293920975110664138, 0.4968941959037306101853293920975110664138]

choose_vector_dyv := [0.006211608192538779629341215804977867172935, 0.5031058040962693898146706079024889335867, 1.0000000000000000000000000000000000000] **(2.3.5)**

```
> choose_vector_dyu_Bernoulli:=subs(x=x0,solution_networks,
solution_derivate_binary_tree,[x*(1+v)^2/dyu, x*D*dyv*(1+v)
/dyu,x*D*(1+v)*dyv/dyu]);
choose_vector_dyu:=CumulativeSum(choose_vector_dyu_Bernoulli);
```

choose_vector_dyu_Bernoulli := [0.004306762784306981741738802945977214579971, 0.4978466186078465091291305985270113927102, 0.4978466186078465091291305985270113927102]

choose_vector_dyu := [0.004306762784306981741738802945977214579971, 0.5021533813921534908708694014729886072902, 1.0000000000000000000000000000000000000] **(2.3.6)**

```
> ch_b_or_dxb_Bernoulli:=subs(x=x0,
solution_derivate_binary_tree,solution_networks,[(u+v)/(u+v+x*
(dxu+dxv)),x*(dxu+dxv)/(u+v+x*(dxu+dxv))]);
ch_b_or_dxb:=CumulativeSum(ch_b_or_dxb_Bernoulli);
```

$ch\_b\_or\_dxb\_Bernoulli := [0.013938163644066635083649672182972999948246,$

$0.9860618363559333649163503278170270005178]$

$ch\_b\_or\_dxb := [0.013938163644066635083649672182972999948246,$ **(2.3.7)**

$1.000000000000000000000000000000000000000]$

```
> ch_3b_or_dyb_Bernoulli:=subs(solution_networks,
  solution_derivate_binary_tree,[3*(u+v)/(3*(u+v)+D*(dyu+dyv)),
  D*(dyu+dyv)/(3*(u+v)+D*(dyu+dyv))]);
  ch_3b_or_dyb:=CumulativeSum(ch_3b_or_dyb_Bernoulli);
```

$ch\_3b\_or\_dyb\_Bernoulli := [0.01702895461509294433999731957283858317434,$

$0.9829710453849070556600026804271614168257]$

$ch\_3b\_or\_dyb := [0.01702895461509294433999731957283858317434,$ **(2.3.8)**

$1.000000000000000000000000000000000000000]$

Calculating the choose-vectors of 3-connected networks

```
> ch_K_in_dyK:=subs(y=y0,solution_networks,
  solution_derivate_3_connected,Array([K/(y*dyK),1]));
```

$ch\_K\_in\_dyK := \left[\ 0.09883853176333715697807141072472118303137 \quad 1\ \right]$ **(2.4.1)**

```
> ch_dxK_in_dxyK:=subs(y=y0,solution_bi_derivate_3_connected,
  solution_derivate_3_connected,Array([dxK/(y*dxyK),1]));
```

$ch\_dxK\_in\_dxyK := \left[\ 0.003397034693621610456156423239028744632855 \quad 1\ \right]$ **(2.4.2)**

Calculating the choose-vectors of networks

```
> choose_vector_non_trivial_D_Bernoulli:=subs(y=y0,
  solution_networks,[S/(D-y),P/(D-y),H/(D-y)]);
  choose_vector_non_trivial_D:=CumulativeSum
  (choose_vector_non_trivial_D_Bernoulli);
```

$choose\_vector\_non\_trivial\_D\_Bernoulli :=$

$[0.4660448236292321853614485334179461733415,$

$0.5114140646637983264311148619749228459781,$

$0.02254111170696948820743660460713098068369]$

$choose\_vector\_non\_trivial\_D := [0.4660448236292321853614485334179461733415,$ **(2.5.1)**

$0.9774588882930305117925633953928690193196,$

$1.0000000000000000000000000000000000000003]$

```
> choose_vector_D_Bernoulli:=subs(y=y0,solution_networks,[y/D,
  S/D,P/D,H/D]);
  choose_vector_D:=CumulativeSum(choose_vector_D_Bernoulli);
```

$choose\_vector\_D\_Bernoulli := [0.9139366196864386281104912070364764757454,$

$0.04010939289916924293617899073458986103253,$

0.0440140231448647433305756468567058659570 1,

0.0019399642695273856227541553722277972653 20]

choose_vector_D := [0.9139366196864386281104912070364764757454, **(2.5.2)**

0.9540460125856078710466701977710663367779,

0.9980600357304726143772458446277722027349,

1.0000000000000000000000000000000000000000]

```
> choose_vector_P_Bernoulli:=evalf(subs(y=y0,solution_networks,
  [y*(exp(S+H)-1)/P,(exp(S+H)-S-H-1)/P]));
  choose_vector_P:=CumulativeSum(choose_vector_P_Bernoulli);
```

choose_vector_P_Bernoulli := [0.9776813633952326097847671846753714434 8,

0.0223186366047673902152328153246285565 1]

choose_vector_P := [0.97768136339523260978476718467537144348, **(2.5.3)**

0.99999999999999999999999999999999999999]

```
> ch_y_or_P_or_H_Bernoulli:=subs(y=y0,solution_networks,[y/(y+P+
  H),P/(y+P+H),H/(y+P+H)]);
  ch_y_or_P_or_H:=CumulativeSum(ch_y_or_P_or_H_Bernoulli);
```

ch_y_or_P_or_H_Bernoulli := [0.9521258078009664929348742134536965641811,

0.0458531657870898760763584370675524328530 2,

0.0020210264119436309887673494787510029659 20]

ch_y_or_P_or_H := [0.9521258078009664929348742134536965641811, **(2.5.4)**

0.9979789735880563690112326505212489970341,

1.0000000000000000000000000000000000000000]

```
> ch_S_or_H_Bernoulli:=subs(solution_networks,[S/(S+H),H/(S+H)])
  ;
  ch_S_or_H:=CumulativeSum(ch_S_or_H_Bernoulli);
```

ch_S_or_H_Bernoulli := [0.9538645915145742239513048725846731731260,

0.0461354084854257760486951274153268268741 4]

ch_S_or_H := [0.9538645915145742239513048725846731731260, **(2.5.5)**

1.0000000000000000000000000000000000000000]

We compute the vector for the Poisson laws of parameter S+H. Notice that the 1+... is only used to have an output without exponent notation. When put in the file, the first 1 has to be replaced by a 0.

```
> exp_S_plus_H:=subs(solution_networks,Poisson(S+H));
  poisson_S_plus_H:=Array([seq(CDF(exp_S_plus_H,k),k=0..19)]);
```

exp_S_plus_H := *Poisson*(0.046009051681421070449606065912777356025 02)

poisson_S_plus_H := [0.9550333174942537091195158934837490645331, **(2.5.6)**

0.9989734947563258457513077193362625776612,

0.9999843176995965857141265765023443379687,

0.9999998200346091555920584564873877009568,

0.9999999983465423495997768478409590294119,

0.9999999999873349395472173280131206824994,

0.9999999999999168247254480812463660010632,

0.9999999999999995219547930704580375483816,

0.9999999999999999975574304259505097640241,

0.9999999999999999999887666839404004906597,

0.9999999999999999999999530314952774702066,

0.9999999999999999999999998199718937132226,

0.9999999999999999999999999993630136075298,

0.9999999999999999999999999999979070926379,

0.9999999999999999999999999999999935817213,

0.9999999999999999999999999999999999815469,

0.9999999999999999999999999999999999999501,

0.9999999999999999999999999999999999999999,

1.000000000000000000000000000000000000000,

1.000000000000000000000000000000000000000]

```
> poisson_at_least_1_S_plus_H:=subs(p0=ProbabilityFunction
  (exp_S_plus_H,0),Array([seq((CDF(exp_S_plus_H,k)-p0)/(1-p0),k=
  1..20)]));
```

poisson_at_least_1_S_plus_H := [0.9771718706724033471377702760501543080471, **(2.5.7)**

0.9996512462221021031467030187646168163116,

0.9999959978059128242300250091371072822056,

0.9999999632292720240385122179735704297386,

0.9999999997183456784661798359796591979374,

0.9999999999981502910618035967400777007521,

0.9999999999999893689020338902568327243432,

0.9999999999999999456804585542339186277748,

0.9999999999999999997501857946010581445496,

0.9999999999999999999989554821012973841305,

0.9999999999999999999999959964112037001704,

0.9999999999999999999999999858342586783270,

0.9999999999999999999999999999534564872151,

0.9999999999999999999999999999998572659057,

0.9999999999999999999999999999999995896273,

0.9999999999999999999999999999999999988903,

0.9999999999999999999999999999999999999978,

1.0000000000000000000000000000000000000000,

1.0000000000000000000000000000000000000000,

1.0000000000000000000000000000000000000000]

```
> poisson_at_least_2_S_plus_H:=subs(p0=ProbabilityFunction
  (exp_S_plus_H,0),p1=ProbabilityFunction(exp_S_plus_H,1),Array(
  [seq((CDF(exp_S_plus_H,k)-p0-p1)/(1-p0-p1),k=2..21)]));
```

poisson_at_least_2_S_plus_H := [0.9847226300108483888314703939275432426046, **(2.5.8)**

0.9998246814695263897357899406538699102300,

0.9999983892360408388875755112165847544197,

0.9999999876619622443906685302633998381181,

0.9999999999189723822021495112478799375127,

0.9999999999995342983293310005991396390848,

0.9999999999999976204996622259434738212995,

0.9999999999999999890567377723349216103655,

0.9999999999999999999542442622558690931131,

0.9999999999999999999998246203734504018704,

0.9999999999999999999999993794611411917931,

0.9999999999999999999999999979611332966903,

0.9999999999999999999999999999937474467475,

0.9999999999999999999999999999999820233748,

0.9999999999999999999999999999999999513885,

0.9999999999999999999999999999999999999026, 1, 1, 1, 1]

Calculating the choose-vectors of derivate networks

```
> choose_vector_dD_Bernoulli:=subs(solution_derivate_networks,
  [dS/dD,dP/dD,dH/dD]);
  choose_vector_dD:=CumulativeSum(choose_vector_dD_Bernoulli);
```

choose_vector_dD_Bernoulli := [0.3862963395495335154733611776464861932390,

0.5224833412528731454402420532581254677334,

0.09122031919759333908639676909538833902759]

$choose\_vector\_dD$:= [0.3862963395495335154733611776464861932390, **(2.6.1)**

0.9087796808024066609136032309046116609724,

1.00000000000000000000000000000000000000]

```
> choose_vector_dS_Bernoulli:=subs(x=x0,y=y0,solution_networks,
  solution_derivate_networks,[(dP+dH)*x*D/dS,(y+P+H)*D/dS,(y+P+
  H)*x*dD/dS]);
  choose_vector_dS:=CumulativeSum(choose_vector_dS_Bernoulli);
```

$choose\_vector\_dS\_Bernoulli$:= [0.06638385475909259006305056762499804992803,

0.8297855136953394076237726209123564069185,

0.1038306315455680023131768114626455431537]

$choose\_vector\_dS$:= [0.06638385475909259006305056762499804992803, **(2.6.2)**

0.8961693684544319976868231885373544568465,

1.00000000000000000000000000000000000000]

```
> choose_vector_dP_Bernoulli:=evalf(subs(y=y0,solution_networks,
  solution_derivate_networks,[y*(dS+dH)*exp(S+H)/dP,(dS+dH)*(exp
  (S+H)-1)/dP]));
  choose_vector_dP:=CumulativeSum(choose_vector_dP_Bernoulli);
```

$choose\_vector\_dP\_Bernoulli$:= [0.9569683098432193140552456035182382378728,

0.0430316901567806859447543964817617621273]

$choose\_vector\_dP$:= [0.9569683098432193140552456035182382378728, **(2.6.3)**

1.00000000000000000000000000000000000000]

```
> choose_vector_dH_Bernoulli:=subs
  (solution_derivate_3_connected,solution_derivate_networks,
  [dxK/dH,dD*dyK/dH]);
  choose_vector_dH:=CumulativeSum(choose_vector_dH_Bernoulli);
```

$choose\_vector\_dH\_Bernoulli$:= [0.7645710969244040241448556090942396686980,

0.2354289030755959758551443909057603313019]

$choose\_vector\_dH$:= [0.7645710969244040241448556090942396686980, **(2.6.4)**

0.99999999999999999999999999999999999999]

```
> ch_dP_or_dH_Bernoulli:=subs(solution_derivate_networks,[dP/
  (dP+dH),dH/(dP+dH)]);
  ch_dP_or_dH:=CumulativeSum(ch_dP_or_dH_Bernoulli);
```

$ch\_dP\_or\_dH\_Bernoulli$:= [0.8513609660880359813779854493338720088336,

0.1486390339119640186220145506661279911663]

$ch\_dP\_or\_dH$:= [0.8513609660880359813779854493338720088336, **(2.6.5)**

0.99999999999999999999999999999999999999]

```
> ch_dS_or_dH_Bernoulli:=subs(solution_derivate_networks,[dS/
```

```
(dS+dH),dH/(dS+dH)]);
ch_dS_or_dH:=CumulativeSum(ch_dS_or_dH_Bernoulli);
```

ch_dS_or_dH_Bernoulli := [0.80896934687696443022499331790351812910 48,

0.19103065312303556977500668209648187089 51]

ch_dS_or_dH := [0.80896934687696443022499331790351812910 48, **(2.6.6)**

0.99999999999999999999999999999999999999 9]

Calculating the choose-vectors of bi-derivate networks

```
> choose_vector_ddD_Bernoulli:=subs
  (solution_bi_derivate_networks,[ddS/ddD,ddP/ddD,ddH/ddD]);
  choose_vector_ddD:=CumulativeSum(choose_vector_ddD_Bernoulli);
```

choose_vector_ddD_Bernoulli := [0.081776877787828180239810135548904017878 57,

0.52323011158842108821746692711862938030 14,

0.39499301062375073154272293733246660182 02]

choose_vector_ddD := [0.081776877787828180239810135548904017878 57, **(2.7.1)**

0.60500698937624926845727706266753339818 00,

1.00000000000000000000000000000000000000 0]

```
> choose_vector_ddS_Bernoulli:=subs(x=x0,y=y0,solution_networks,
  solution_derivate_networks,solution_bi_derivate_networks,[
  (ddP+ddH)*x*D,2*(dP+dH)*D,2*(dP+dH)*x*dD,2*(y+P+H)*dD,(y+P+H)*
  x*ddD]/ddS);
  choose_vector_ddS:=CumulativeSum(choose_vector_ddS_Bernoulli);
```

choose_vector_ddS_Bernoulli := [0.46918276027935104811035987555843869105 06,

0.015001993271762265826772293051033065070 86,

0.0018771916478905226612868970429549244216 12,

0.023464537296037100408402184734079106142 66,

0.49047351750495906299317874961349421331 48]

choose_vector_ddS := [0.46918276027935104811035987555843869105 06, **(2.7.2)**

0.48418475355111331393713216860947175612 15,

0.48606194519900383659841906565242668054 31,

0.50952648249504093700682125038650578668 58,

1.00000000000000000000000000000000000000 01]

```
> choose_vector_ddP_Bernoulli:=evalf(subs(x=x0,y=y0,
  solution_networks,solution_derivate_networks,
  solution_bi_derivate_networks,[y*(ddS+ddH)*exp(S+H),y*(dS+dH)
  ^2*exp(S+H),(ddS+ddH)*(exp(S+H)-1),(dS+dH)^2*exp(S+H)]/ddP));
  choose_vector_ddP:=CumulativeSum(choose_vector_ddP_Bernoulli);
```

choose_vector_ddP_Bernoulli := [0.95410806403610963694261952398038220225 96,

0.0014944307861031850052121952190199419411 31,

0.0429030743916839930469560855815779138585,

0.0014944307861031850052121952190199419411 31]

choose_vector_ddP := [0.9541080640361096369426195239803822022596, **(2.7.3)**

0.9556024948222128219478317191994021442007,

0.9985055692138968149947878047809800580592,

1.0000000000000000000000000000000000000000]

```
> choose_vector_ddH_Bernoulli:=evalf(subs
  (solution_derivate_3_connected,solution_derivate_networks,
  solution_bi_derivate_3_connected,
  solution_bi_derivate_networks,[dxxK,2*dD*dxyK,ddD*dyK,dD^2*
  dyyK]/ddH));
  choose_vector_ddH:=CumulativeSum(choose_vector_ddH_Bernoulli);
```

choose_vector_ddH_Bernoulli := [0.5533555720999897237012340568654618818829,

0.3404552770598381731840867021260544804365,

0.0543703283584223351724458569788730005197 0,

0.0518188224817497679422333840296106371604 9]

choose_vector_ddH := [0.5533555720999897237012340568654618818829, **(2.7.4)**

0.8938108491598278968853207589915163623194,

0.9481811775182502320577666159703893628391,

0.9999999999999999999999999999999999999996]

```
> ch_ddP_or_ddH_Bernoulli:=subs(solution_bi_derivate_networks,
  [ddP,ddH]/(ddP+ddH));
  ch_ddP_or_ddH:=CumulativeSum(ch_ddP_or_ddH_Bernoulli);
```

ch_ddP_or_ddH_Bernoulli := [0.5698289434575134172367965799050243638847,

0.4301710565424865827632034200949756361152]

ch_ddP_or_ddH := [0.5698289434575134172367965799050243638847, **(2.7.5)**

0.9999999999999999999999999999999999999999]

```
> ch_ddS_or_ddH_Bernoulli:=subs(solution_bi_derivate_networks,
  [ddS,ddH]/(ddS+ddH));
  ch_ddS_or_ddH:=CumulativeSum(ch_ddS_or_ddH_Bernoulli);
```

ch_ddS_or_ddH_Bernoulli := [0.1715227403733036038782044215405283009794,

0.8284772596266963961217955784594716990203]

ch_ddS_or_ddH := [0.1715227403733036038782044215405283009794, **(2.7.6)**

0.9999999999999999999999999999999999999997]

Evaluation of the generating function of 2-connected planar graphs

```
> beta_1:=z*(6*x-2+x*z)/(4*x)+(1+z)*ln((1+y)/(1+z))-ln(1+z)/2+ln
  (1+x*z)/(2*x^2):
  beta_2:=(2*(1+x)*(1+w)*(z+w^2)+3*(w-z))/(2*(1+w)^2)-1/(2*x)*ln
  (1+x*z+x*w+x*w^2)+(1-4*x)/(2*x)*ln(1+w)+(1-4*x+2*x^2)/(4*x)*ln
  ((1-x+x*z-x*w+x*w^2)/((1-x)*(z+w^2+1+w))):
  B:=subs(z=D,w=D*(1+u),x^2/2*beta_1-x/4*beta_2);
```

$$B := \frac{1}{2}x^2\left(\frac{1}{4}\frac{D(6x-2+xD)}{x}+(1+D)\ln\left(\frac{1+y}{1+D}\right)-\frac{1}{2}\ln(1+D)+\frac{1}{2}\frac{\ln(1+xD)}{x^2}\right)-\frac{1}{4}x\left(\frac{1}{2}\frac{1}{(1+D(1+u))^2}\left(2(1+x)(1+D(1+u))\left(D+D^2(1+u)^2\right)+3D(1+u)-3D\right)-\frac{1}{2}\frac{\ln\left(1+xD+xD(1+u)+xD^2(1+u)^2\right)}{x}+\frac{1}{2}\frac{(1-4x)\ln(1+D(1+u))}{x}+\frac{1}{4}\frac{\left(1-4x+2x^2\right)\ln\left(\frac{1-x+xD-xD(1+u)+xD^2(1+u)^2}{(1-x)\left(D+D^2(1+u)^2+1+D(1+u)\right)}\right)}{x}\right) \tag{2.8.1}$$

```
> eval_2connected := evalf(subs(x=x0,y=y0,eval_networks(x0,y0),
  B));
```

$$eval\_2connected := 0.000739625001480403343297708612043540283 74 \tag{2.8.2}$$

Evaluation of the generating function of derivate 2-connected planar graphs

```
> dBx:=subs({DISS(x,y)=D,diff(DISS(x,y),x)=dD,u(x,DISS(x,y))=u,D
  [1](u)(x,DISS(x,y))=dxu,D[2](u)(x,DISS(x,y))=dyu},diff(subs(
  {D=DISS(x,y),u=u(x,DISS(x,y))},B),x)):
> eval_derivate_2connected := evalf(subs(x=x0, y=y0,
  eval_networks(x0, y0), solution_derivate_binary_tree,
  solution_derivate_networks, solution_derivate_3_connected,
  dBx));
```

(2.9.1)

$$eval\_derivate\_2connected := 0.03904979412040430977059064390750957114218 \quad (2.9.1)$$

Evaluation of the generating function of bi-derivate 2-connected planar graphs

```
> dBxx:=subs({DISS(x,y)=D,diff(DISS(x,y),x)=dD,diff(DISS(x,y),x,
  x)=ddD,u(x,DISS(x,y))=u,D[1](u)(x,DISS(x,y))=dxu,D[2](u)(x,
  DISS(x,y))=dyu,D[1,1](u)(x,DISS(x,y))=dxxu,D[1,2](u)(x,DISS(x,
  y))=dxyu,D[2,2](u)(x,DISS(x,y))=dyyu},diff(subs({D=DISS(x,y),
  dD=diff(DISS(x,y),x),u=u(x,DISS(x,y)),dxu=D[1](u)(x,DISS(x,y)
  ),dyu=D[2](u)(x,DISS(x,y))},dBx),x)):
> eval_bi_derivate_2connected := evalf(subs(x=x0,y=y0,
  solution_networks,solution_derivate_networks,
  solution_derivate_binary_tree,solution_bi_derivate_networks,
  solution_bi_derivate_binary_tree, dBxx));
```

$$eval\_bi\_derivate\_2connected := 1.0518999270653559878545196836236520068 44 \quad (2.10.1)$$

Evaluation of the generating function of tri-derivate 2-connected planar graphs

```
> dBxxx:=subs({DISS(x,y)=D,diff(DISS(x,y),x)=dD,diff(DISS(x,y),
  x,x)=ddD,diff(DISS(x,y),x,x,x)=dddD,u(x,DISS(x,y))=u,D[1](u)
  (x,DISS(x,y))=dxu,D[2](u)(x,DISS(x,y))=dyu,D[1,1](u)(x,DISS(x,
  y))=dxxu,D[1,2](u)(x,DISS(x,y))=dxyu,D[2,2](u)(x,DISS(x,y))=
  dyyu,D[1,1,1](u)(x,DISS(x,y))=dxxxu,D[1,1,2](u)(x,DISS(x,y))=
  dxxyu,D[1,2,2](u)(x,DISS(x,y))=dxyyu,D[2,2,2](u)(x,DISS(x,y))=
  dyyyu},diff(subs({D=DISS(x,y),dD=diff(DISS(x,y),x),ddD=diff
  (DISS(x,y),x,x),u=u(x,DISS(x,y)),dxu=D[1](u)(x,DISS(x,y)),dyu=
  D[2](u)(x,DISS(x,y)),dxxu=D[1,1](u)(x,DISS(x,y)),dxyu=D[1,2]
  (u)(x,DISS(x,y)),dyyu=D[2,2](u)(x,DISS(x,y))},dBxx),x)):
> eval_tri_derivate_2connected := evalf(subs(x=x0, y=y0,
  solution_networks, solution_derivate_networks,
  solution_bi_derivate_networks, solution_tri_derivate_networks,
  solution_derivate_binary_tree,
  solution_bi_derivate_binary_tree,
  solution_tri_derivate_binary_tree, dBxxx));
```

$$eval\_tri\_derivate\_2connected := 18.325626120319479044893989330769293056 23 \quad (2.11.1)$$

Choose vectors for 2 connected

```
> ch_xy_in_dB := subs(x=x0, y=y0, dB=eval_derivate_2connected
  (x0,y0), Array([x*y/dB, 1]));
```

$$ch\_xy\_in\_dB := \left[\begin{array}{cc} 0.977961317715968165733590287843393373720 9 & 1 \end{array}\right] \quad (2.12.1)$$

```
> ch_y_in_ddB := subs(x=y0, y=y0, ddB=
  eval_bi_derivate_2connected, Array([y/ddB, 1]));
```

$$ch\_y\_in\_ddB := \left[\begin{array}{cc} 0.950660775108000008377482550274744340014 4 & 1 \end{array}\right] \quad (2.12.2)$$

```
> ch_nontrivialD_or_dD := subs(x=x0, y=y0, solution_networks,
  solution_derivate_networks, Array([(D-y)/(x * dD + D - y), 1])
  );
```

$ch\_nontrivialD\_or\_dD := \begin{bmatrix} 0.40751083795296888949982189069815932444452 & 1 \end{bmatrix}$ **(2.12.3)**

```
> ch_dD_or_ddD := subs(x=x0, solution_derivate_networks,
  solution_bi_derivate_networks, Array([dD/(dD+x*ddD),1]));
```

$ch\_dD\_or\_ddD := \begin{bmatrix} 0.02336147645348866469567040469843764424012 & 1 \end{bmatrix}$ **(2.12.4)**

All together

```
> f:=fopen("/tmp/values_networks",WRITE):


  fprintf(f,"%{}a\n",ch_1_or_u):
  fprintf(f,"%{}a\n",ch_1_or_v):
  fprintf(f,"%{}a\n",ch_u_or_v):
  fprintf(f,"%{}a\n",ch_dxu_or_dxv):
  fprintf(f,"%{}a\n",choose_vector_dxu):
  fprintf(f,"%{}a\n",choose_vector_dxv):
  fprintf(f,"%{}a\n",ch_dyu_or_dyv):
  fprintf(f,"%{}a\n",choose_vector_dyv):
  fprintf(f,"%{}a\n",choose_vector_dyu):

  fprintf(f,"%{}a\n",ch_K_in_dyK):
  fprintf(f,"%{}a\n",ch_dxK_in_dxyK):
  fprintf(f,"%{}a\n",ch_b_or_dxb):
  fprintf(f,"%{}a\n",ch_3b_or_dyb):


  fprintf(f,"%{}a\n",choose_vector_non_trivial_D):
  fprintf(f,"%{}a\n",choose_vector_D):
  fprintf(f,"%{}a\n",choose_vector_P):
  fprintf(f,"%{}a\n",ch_y_or_P_or_H):
  fprintf(f,"%{}a\n",ch_S_or_H):
  fprintf(f,"%{}a\n",poisson_S_plus_H):
  fprintf(f,"%{}a\n",poisson_at_least_1_S_plus_H):
  fprintf(f,"%{}a\n",poisson_at_least_2_S_plus_H):
  fprintf(f,"%{}a\n",choose_vector_dD):
  fprintf(f,"%{}a\n",choose_vector_dS):
  fprintf(f,"%{}a\n",choose_vector_dP):
  fprintf(f,"%{}a\n",choose_vector_dH):
  fprintf(f,"%{}a\n",ch_dP_or_dH):
  fprintf(f,"%{}a\n",ch_dS_or_dH):
  fprintf(f,"%{}a\n",choose_vector_ddD):
  fprintf(f,"%{}a\n",choose_vector_ddS):
  fprintf(f,"%{}a\n",choose_vector_ddP):
  fprintf(f,"%{}a\n",choose_vector_ddH):
  fprintf(f,"%{}a\n",ch_ddP_or_ddH):
```

```
fprintf(f,"%{}a\n",ch_ddS_or_ddH):

fprintf(f,"%{}a\n",ch_xy_in_dB):
fprintf(f,"%{}a\n",ch_y_in_ddB):
fprintf(f,"%{}a\n",ch_nontrivialD_or_dD):
fprintf(f,"%{}a\n",ch_dD_or_ddD):


fclose(f):
```

Connected planar graphs

Evaluation of the generating function of pointed connected planar graphs

```
> inverse_F:=proc(x0,y0)
  evalf(subs(system_networks, system_derivate_networks,
  system_3_connected, system_derivate_3_connected,
  system_derivate_binary_tree, x=x0, y=y0, eval_networks(x0,y0),
  x*exp(-dBx)));
  end proc;
```

inverse_F := **proc**(*x0*, *y0*) **(3.1.1)**

evalf(*subs*(*system_networks*, *system_derivate_networks*, *system_3_connected*, *system_derivate_3_connected*, *system_derivate_binary_tree*, $x = x0$, $y = y0$, *eval_networks*(*x0*, *y0*), $x * \exp(-dBx)$))

end proc

```
> inverse_F(x0,y0);
```

0.03672839495542461891883081627184123284976 **(3.1.2)**

```
> eval_F:=proc(z,y0)
  fsolve(x -> inverse_F(x, y0) - z);
  end proc;
```

eval_F := **proc**(*z*, *y0*) *fsolve*($x \rightarrow$ *inverse_F*(*x*, *y0*) $- z$) **end proc** **(3.1.3)**

```
> eval_F(.03672841258183,1);
```

Warning, computation interrupted

```
> inverse_F(.03819109766940242994680766196867813479850,1);
```

0.03672841258182999999999999999999998042413 **(3.1.4)**

Evaluation of the generating function of connected planar graphs

```
> eval_C:=proc(z,y)
  local value_F:

  value_F:=eval_F(z,y):
  value_F*log(z)-value_F*log(value_F)+value_F+eval_2connected
  (value_F,y):
```

```
end proc;
```

$$eval\_C := \mathbf{proc}(z, y) \tag{3.2.1}$$
local *value_F*;
value_F := *eval_F*(*z*, *y*);
value_F * log(*z*) − *value_F* * log(*value_F*) + *value_F* + *eval_2connected*(*value_F*, *y*)
end proc

Evaluation of the generating function of bi-derivate connected planar graphs

```
> eval_derivate_inverse_F:=proc(x0,y0)
    evalf(exp(-eval_derivate_2connected(x0,y0))*(1-x0*
  eval_bi_derivate_2connected(x0,y0))):
  end proc:
> eval_bi_derivate_connected:=proc(z0,y0)
  local F,dC,dF:

  F:=eval_F(z0,y0):
  dF:=evalf(1/eval_derivate_inverse_F(F,y0)):
  evalf(-F/z0^2+dF/z0):

  end proc:
```

Evaluation of the generating function of tri-derivate connected planar graphs

```
> eval_bi_derivate_inverse_F:=proc(x0,y0)
    evalf(exp(-eval_derivate_2connected(x0,y0))*(-2*
  eval_bi_derivate_2connected(x0,y0)+x0*
  eval_bi_derivate_2connected(x0,y0)^2-x0*
  eval_tri_derivate_2connected(x0,y0))):
  end proc:
> eval_tri_derivate_connected:=proc(z0,y0)
  local F,dC,ddF,dF:

  F:=eval_F(z0,y0):
  dF:=evalf(1/eval_derivate_inverse_F(F,y0)):
  ddF:=evalf(-eval_bi_derivate_inverse_F(F,y0)
  /eval_derivate_inverse_F(F,y0)^3):
  evalf(ddF/z0-2*dF/z0^2+2*F/z0^3):

  end proc:
```

Planar graphs and connected planar graphs together

We want to make the expensive calculation of the inverse just once, so we derive from one evaluation of F all the generating functions of connected and planar graphs

```
> evaluate_planar_and_connected:=proc(z0,F,y0)
  local dF,ddF,Ceval,dCeval,ddCeval,dddCeval,Geval,dGeval,ddGeval,
  dddGeval:
  dF:=evalf(1/eval_derivate_inverse_F(F,y0)):
  ddF:=evalf(-eval_bi_derivate_inverse_F(F,y0)
  /eval_derivate_inverse_F(F,y0)^3):
  dCeval:=evalf(F/z0):
  Ceval:=F*log(z0)-F*log(F)+F+eval_2connected(F,y0):
  ddCeval:=evalf(-F/z0^2+dF/z0):
  dddCeval:=evalf(ddF/z0-2*dF/z0^2+2*F/z0^3):
  Geval:=evalf(exp(Ceval)):
  dGeval:=evalf(dCeval*Geval):
  ddGeval:=evalf(ddCeval*Geval+dCeval*dGeval):
  dddGeval:=evalf(dddCeval*Geval+2*ddCeval*dGeval+dCeval*ddGeval):
  {C=Ceval,dC=dCeval,ddC=ddCeval,dddC=dddCeval,G=Geval,dG=dGeval,
  ddG=ddGeval,dddG=dddGeval}:
  end proc:
> evaluate_planar_and_connected(.03672841258183,
  .03819109766940242994680766196867816797991,1);
```

$$\{C = 0.03743936602468554849080543522768041419953,\ G = 1.038149048069666547989910110165655146797,\ dC = 1.039824348093280741233629384704015601806,\ dG = 1.079492657132700989555944229432597122144,\ ddC = 1.185032516943684908486205452044418956770,\ ddG = 2.352723127871181713225745508409676973993,\ dddC = 3.103702933181473985927846514846325896096\ 10^5,\ dddG = 3.222156294439316009246351223141395037544\ 10^5\} \tag{4.1}$$

Calculating all the choose-vectors

Choice of the number of vertices and possibly of balance edges-vertices

N is the wanted average number of vertices

```
> N:=10000: mu:=2: y0:=1:
```

Finding the good z from singularity of generating functions of planar graphs

```
> h_t:=t^2*(1-t)*(18+36*t+5*t^2)/(2*(3+t)*(1+2*t)*(1+3*t)^2):
  y0_t:=(1+2*t)/((1+3*t)*(1-t))*exp(-h_t)-1:
  rho:=-1/16*sqrt(1+3*t)*(-1+t)^3/t^3*exp(1/16*ln(1+t)*(3*t-1)*
  (1+t)^3/t^3-1/32*ln(1+2*t)*(1+3*t)*(-1+t)^3/t^3-1/64*(-1+t)*
```

```
(185*t^4+698*t^3-217*t^2-160*t+6)/t/(1+3*t)^2/(3+t)):
> find_singularity_connected:=proc(y0)
  evalf(subs(t=fsolve(y0_t=y0,t), rho));
  end proc;
```

$$find\_singularity\_connected := \mathbf{proc}(y0)\ evalf(subs(t = fsolve(y0\_t = y0, t), \rho))\ \mathbf{end\ proc} \quad (5.2.1)$$

```
> Rc:=find_singularity_connected(1);
```

$$Rc := 0.036728412581838220293476617184035408144720 \quad (5.2.2)$$

```
> 1/%;
```

$$27.226877768588576467079458051494458287520 \quad (5.2.3)$$

```
> z0:=evalf(Rc*(1-1/(2*N)));
```

$$z0 := 0.036726576161209128382461943353176206374310 \quad (5.2.4)$$

If a particular balance edge-vertices is wanted, execute this part to find the good y from the balance edges-vertices

```
> rho_t:=-1/16*sqrt(1+3*t)*(-1+t)^3*t^(-3)*exp(A);
```

$$rho\_t := -\frac{1}{16}\,\frac{\sqrt{1+3\,t}\,(-1+t)^3\,e^A}{t^3} \quad (5.3.1)$$

```
> A:=log(1+t)*(3*t-1)*(1+t)^3/(16*t^3)-log(1+2*t)*(1+3*t)*(-1+t)
  ^3/(32*t^3)-(-1+t)*(185*t^4+698*t^3-217*t^2-160*t+6)/(64*t*
  (1+3*t)^2*(3+t));
```

$$A := \frac{1}{16}\,\frac{\ln(1+t)\,(3\,t-1)\,(1+t)^3}{t^3} - \frac{1}{32}\,\frac{\ln(1+2\,t)\,(1+3\,t)\,(-1+t)^3}{t^3} - \frac{1}{64}\,\frac{(-1+t)\,(185\,t^4+698\,t^3-217\,t^2-160\,t+6)}{t\,(1+3\,t)^2\,(3+t)} \quad (5.3.2)$$

```
> Y:=(1+2*t)/(1+3*t)/(1-t)*exp(-t^2*(1-t)*(18+36*t+5*t^2)/(2*(3+
  t)*(1+2*t)*(1+3*t)^2))-1;
```

$$Y := \frac{(1+2\,t)\,e^{-\frac{1}{2}\,\frac{t^2\,(1-t)\,(18+36\,t+5\,t^2)}{(3+t)\,(1+2\,t)\,(1+3\,t)^2}}}{(1+3\,t)\,(1-t)} - 1 \quad (5.3.3)$$

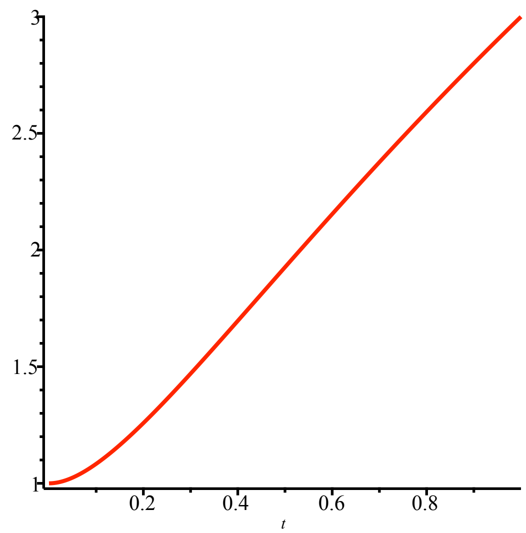
```
> rho_subt_prime:=simplify(diff(rho_t, t)):
  Y_prime:=simplify(diff(Y, t)):
  rho_prime_t:=simplify( rho_subt_prime*1/Y_prime ):
  mu_t:=simplify(-Y*rho_prime_t/rho_t):
  t_mu:=mu->fsolve(mu_t=mu, t, 0..1):
  y_from_expected_edges:=expect_edges->evalf(subs(t=t_mu
  (expect_edges),Y)):
  y_from_expected_edges(1.1);
```

```
y_from_expected_edges(2);
y_from_expected_edges(2.2);
y_from_expected_edges(2.9);
```
0.012966814282347086203536165759856380032

0.613408306229252191639801077529145495803

0.969701292577611404868130565290704156376

13.80786861666055582292203951666798051008 **(5.3.4)**

```
> plot(mu_t, t=0+0.0001..1-0.0001);
```



```
> y0:=y_from_expected_edges(mu):
```

```
> x0:=eval_F(z0,y0);
```
Warning, computation interrupted

Calculating the choose-vectors of bicolored binary trees

```
> solution_networks:=eval_networks(x0,y0);
```
solutions_networks $:=$ {K = 0.002122620261197028148869821422589998664495, P = 0.04815857618724628897480927684861108580802, S = 0.04388629460482507723993229504399076251673, H = 0.002122620261197028148869821422589998664495, D = 1.094167491053268394363611393315191846989, u = 0.5254575010528156100642479769635852069262, v = 2.546150277874201904358407655542267890527} **(5.4.1)**

```
> u_eval:=subs(solution_networks,u):v_eval:=subs
  (solution_networks,v):
```

```
> ch_1_or_u:=[evalf(1/(1+u_eval))];
```
ch_1_or_u $:=$ [0.6555410421528204868268866226466062451797] **(5.4.2)**

```
> ch_1_or_v:=[evalf(1/(1+v_eval))];
```
ch_1_or_v $:=$ [0.2819959453606310344033925095368809490853] **(5.4.3)**

```
> ch_u_or_v:=[evalf(u_eval/(u_eval+v_eval))];
```
$$ch\_u\_or\_v := [0.1710692050781203161735415238933961027360] \quad (5.4.4)$$

Calculating the choose-vectors of pointed bicolored binary trees

```
> solution_derivate_binary_tree:=subs({x=x0,op
  (solution_networks)},system_derivate_binary_tree);
```
$$solution\_derivate\_binary\_tree := \{dxu = 1284.925949185499208199749498632786654547, dxv = 4289.355239734284787779411716905650192180, dyu = 109.2477594753085330778871502648066881920, dyv = 367.0192036909007077466745649307396860163\} \quad (5.5.1)$$

Here pU and pV stand for the generating functions of black-rooted and white-rooted bicolored binary trees with a pointed black vertex

```
> dxu_eval:=evalf(subs(solution_derivate_binary_tree,dxu))
  :dxv_eval:=evalf(subs(solution_derivate_binary_tree,dxv))
  :dyu_eval:=evalf(subs(solution_derivate_binary_tree,dyu))
  :dyv_eval:=evalf(subs(solution_derivate_binary_tree,dyv)):
> ch_dxu_or_dxv:=[evalf(dxu_eval/(dxu_eval+dxv_eval))];
```
$$ch\_dxu\_or\_dxv := [0.2305097115910830985084671172699999783268] \quad (5.5.2)$$

```
> D_eval:=subs(solution_networks,D);
```
$$D\_eval := 1.094167491053268394363611393315191846989 \quad (5.5.3)$$

```
> choose_vector_dxu_Bernoulli:=[(1+v_eval)^2*D_eval/dxu_eval,x0*
  D_eval*dxv_eval*(1+v_eval)/dxu_eval,(1+v_eval)*x0*D_eval*
  dxv_eval/dxu_eval];
  choose_vector_dxu:=[choose_vector_dxu_Bernoulli[1],
  choose_vector_dxu_Bernoulli[1]+choose_vector_dxu_Bernoulli[2]]
  ;
```
$$choose\_vector\_dxu\_Bernoulli := [0.01070828643549451802688641337274819561550, 0.4946458567822527409865567933136259021921, 0.4946458567822527409865567933136259021921] \quad (5.5.4)$$

$$choose\_vector\_dxu := [0.01070828643549451802688641337274819561550, 0.5053541432177472590134432066863740978076]$$

```
> choose_vector_dxv:=[0.5];
```
$$choose\_vector\_dxv := [0.5] \quad (5.5.5)$$

```
> ch_dyu_or_dyv:=[evalf(dyu_eval/(dyu_eval+dyv_eval))];
```
$$ch\_dyu\_or\_dyv := [0.2293834507206473691014551584061148930687] \quad (5.5.6)$$

```
> choose_vector_dyv_Bernoulli:=[(1+u_eval)^2/dyv_eval,dyu_eval*
  D_eval*(1+u_eval)/dyv_eval,(1+u_eval)*D_eval*
  dyu_eval/dyv_eval];
  choose_vector_dyv:=[choose_vector_dyv_Bernoulli[1],
```

```
choose_vector_dyv_Bernoulli[1]+choose_vector_dyv_Bernoulli[2]]
;
```

choose_vector_dyv_Bernoulli := [0.0063403237871936819234289354970883569921 76, 0.49682983810640315903828553225145582150 45, 0.49682983810640315903828553225145582150 45] **(5.5.7)**

choose_vector_dyv := [0.0063403237871936819234289354970883569921 76, 0.50317016189359684096171446774854417849 67]

```
> choose_vector_dyu_Bernoulli:=[x0*(1+v_eval)^2/dyu_eval,x0*
  D_eval*dyv_eval*(1+v_eval)/dyu_eval,x0*D_eval*(1+v_eval)*
  dyv_eval/dyu_eval];
  choose_vector_dyu:=[choose_vector_dyu_Bernoulli[1],
  choose_vector_dyu_Bernoulli[1]+choose_vector_dyu_Bernoulli[2]]
  ;
```

choose_vector_dyu_Bernoulli := [0.0043958336570008652822035772634551200093 65, 0.49780208317149956735889821136827243999 51, 0.49780208317149956735889821136827243999 51] **(5.5.8)**

choose_vector_dyu := [0.0043958336570008652822035772634551200093 65, 0.50219791682850043264110178863172756000 45]

```
> ch_b_or_dxb:=[evalf((u_eval+v_eval)/(u_eval+v_eval+x0*
  dxu_eval+x0*dxv_eval))];
```

ch_b_or_dxb := [0.0142238019740558387580644410680526380266 7] **(5.5.9)**

```
> ch_3b_or_dyb:=[evalf((3*u_eval+3*v_eval)/(3*u_eval+3*v_eval+
  D_eval*dyu_eval+D_eval*dyv_eval))];
```

ch_3b_or_dyb := [0.0173756189856057958788968973271619240569 6] **(5.5.10)**

Calculating the choose-vectors of 3-connected networks

```
> solution_derivate_3_connected:=subs({x=x0,op(eval_networks(x0,
  y0))},system_derivate_3_connected);
  solution_bi_derivate_3_connected:=subs({x=x0,op(eval_networks
  (x0,y0))},system_bi_derivate_3_connected);
```

solution_derivate_3_connected := {*dxK* = 0.24995308076741105403549739058663458360 04, *dyK* = 0.02146830580207235924699971455123496234501} **(5.6.1)**

solution_bi_derivate_3_connected := {*dxyK* = 72.098119886688268463880799694539689456 20, *dxxK* = 840.10357287530259037691474829718763007 40, *dyyK*

= 6.1214245874596275841259633503370093579955}

```
> K_eval:=subs(solution_networks,K);dxK_eval:=subs
  (solution_derivate_3_connected,dxK);dyK_eval:=subs
  (solution_derivate_3_connected,dyK);dxyK_eval:=subs
  (solution_bi_derivate_3_connected,dxyK);
```

$$K\_eval := 0.0021226202611970281488698214225899986644495 \quad (5.6.2)$$

$$dxK\_eval := 0.24995308076741105403549739058663458 36004$$

$$dyK\_eval := 0.021468305802072359246999714551234962 34501$$

$$dxyK\_eval := 72.098119886688268463880799694539689 45620$$

```
> ch_K_in_dyK:=[evalf(K_eval/(y0*dyK_eval))];
```

$$ch\_K\_in\_dyK := [0.098872276218094921723042646700265140 61165] \quad (5.6.3)$$

```
> ch_dxK_in_dxyK:=[evalf(dxK_eval/(y0*dxyK_eval))];
```

$$ch\_dxK\_in\_dxyK := [0.0034668460309401324302276685621587 82282799] \quad (5.6.4)$$

Calculating the choose-vectors of networks

We recall the system verified by the networks

```
> system_verified_by_networks:={
  D=y+S+P+H,
  S=(y+P+H)*x*D,
  P=y*(exp(S+H)-1)+(exp(S+H)-S-H-1),
  H=K}:
> solution_networks:=eval_networks(x0,y0);
```

$solutions\_networks := \{K = 0.0021226202611970281488698214225899986644495, P$ **(5.7.1)**

$= 0.048158576187246288974809276848611085808 02, S$

$= 0.043886294604825077239932295043990762516 73, H$

$= 0.0021226202611970281488698214225899986644495$, D

$= 1.0941674910532683943636113933151918469 89, u$

$= 0.5254575010528156100642479769635852069262, v$

$= 2.5461502778742019043584076555422678905 27\}$

```
> Deval:=subs(solution_networks,D):Seval:=subs
  (solution_networks,S):Peval:=subs(solution_networks,P):Heval:=
  subs(solution_networks,H):
> choose_vector_non_trivial_D_Bernoulli:=[evalf(Seval/(Deval-y0)
  ),evalf(Peval/(Deval-y0)),evalf(Heval/(Deval-y0))];
  choose_vector_non_trivial_D:=
  [choose_vector_non_trivial_D_Bernoulli[1],
  choose_vector_non_trivial_D_Bernoulli[1]+
  choose_vector_non_trivial_D_Bernoulli[2]];
```

$choose\_vector\_non\_trivial\_D\_Bernoulli :=$ **(5.7.2)**

$[0.46604506623203547879032731006401855924 06,$

0.51141403098447302465820387549184288627 82,

0.02254090278349149655146881444413855448383]

choose_vector_non_trivial_D := [0.4660450662320354787903273100640185592406,

0.9774590972165085034485311855558614455188]

```
> choose_vector_D_Bernoulli:=[evalf(y0/Deval),evalf
  (Seval/Deval),evalf(Peval/Deval),evalf(Heval/Deval)];
  choose_vector_D:=[choose_vector_D_Bernoulli[1],
  choose_vector_D_Bernoulli[1]+choose_vector_D_Bernoulli[2],
  choose_vector_D_Bernoulli[1]+choose_vector_D_Bernoulli[2]+
  choose_vector_D_Bernoulli[3]];
```

choose_vector_D_Bernoulli := [0.9139368590062744342410859200525609975617, **(5.7.3)**

0.04010930224455783900667900984635503927846,

0.04401389785478623702447472373752202939577,

0.001939940894381489727760346363561933764260]

choose_vector_D := [0.9139368590062744342410859200525609975617,

0.9540461612508322732477649298989160368402,

0.9980600591056185102722396536364380662360]

```
> choose_vector_P:=[evalf(y0*(exp(Seval+Heval)-1))/Peval];
```

choose_vector_P := [0.9776814277807337582047753429048138191305] **(5.7.4)**

```
> ch_y_or_P_or_H_Bernoulli:=[evalf(y0/(y0+Peval+Heval)),evalf
  (Peval/(y0+Peval+Heval)),evalf(Heval/(y0+Peval+Heval))];
  ch_y_or_P_or_H:=[ch_y_or_P_or_H_Bernoulli[1],
  ch_y_or_P_or_H_Bernoulli[1]+ch_y_or_P_or_H_Bernoulli[2]];
```

ch_y_or_P_or_H_Bernoulli := [0.9521259671995741157203691395948569482917, **(5.7.5)**

0.04585303093123625121422342586135633703728,

0.002021001869189633065407434543786714671500]

ch_y_or_P_or_H := [0.9521259671995741157203691395948569482917,

0.9979789981308103669345925654562132853290]

```
> ch_S_or_H:=[evalf(Seval/(Seval+Heval))];
```

ch_S_or_H := [0.9538650223032189450975073317278092068268] **(5.7.6)**

We compute the vector for the Poisson laws of parameter S+H. Notice that the 1+... is only used to have an output without exponent notation. When put in the file, the first 1 has to be replaced by a 0.

```
> exp_S_plus_H:=evalf(exp(Seval+Heval)):
  poisson_S_plus_H:=[seq(0,i=1..21)]:
  poisson_S_plus_H[1]:=evalf(1/exp_S_plus_H):
  for i from 2 to 21 do poisson_S_plus_H[i]:=poisson_S_plus_H
  [i-1]+evalf((Seval+Heval)^(i-1)/(i-1)!/exp_S_plus_H);
```

```
od:
poisson_S_plus_H;
```

[0.9550334481575270053951502366549727789542, **(5.7.7)**

0.9989735007680102013291621746569884681723,

0.9999843178378923281955614210534971206075,

0.9999998200367301044262799250014866620138,

0.9999999983465667452746364296562155442393,

0.9999999999873351640312566417971884601512,

0.9999999999999168264468284829202084153866,

0.9999999999999995219661072079021334691215,

0.9999999999999999975574954949455628470009,

0.9999999999999999999887670165802121631832,

0.9999999999999999999999530330257194233578,

0.9999999999999999999999998199782949930608,

0.9999999999999999999999999993630381505608,

0.9999999999999999999999999999979071794990,

0.9999999999999999999999999999999935820064,

0.9999999999999999999999999999999999815474,

0.9999999999999999999999999999999999999496,

0.9999999999999999999999999999999999999994,

0.9999999999999999999999999999999999999995,

0.9999999999999999999999999999999999999995,

0.9999999999999999999999999999999999999995]

```
> exp_at_least_1_S_plus_H:=evalf(exp(Seval+Heval)-1):
poisson_at_least_1_S_plus_H:=[seq(0,i=1..21)]:
poisson_at_least_1_S_plus_H[1]:=evalf((Seval+Heval)
/exp_at_least_1_S_plus_H):
for i from 2 to 21 do
poisson_at_least_1_S_plus_H[i]:=poisson_at_least_1_S_plus_H
[i-1]+evalf((Seval+Heval)^i/i!/exp_at_least_1_S_plus_H);
od:
poisson_at_least_1_S_plus_H;
```

[0.9771719380310539506835922490163142274384, **(5.7.8)**

0.9996512482842222455070277752132408328031,

0.9999959978414505514727594363181515406030,

0.9999999632297077054589908737677468542814,

0.9999999997183498522832957585512522730404,

0.9999999999981503239682933437721104538053,

0.9999999999999893691227544698529755475621,

0.9999999999999999456817477664058216948548,

0.9999999999999999997501924661882176435655,

0.9999999999999999999989555131012688823577,

0.9999999999999999999999959965419265948626,

0.9999999999999999999999999858347633220674,

0.9999999999999999999999999999534582836531,

0.9999999999999999999999999999998572718322,

0.9999999999999999999999999999999995896387,

0.9999999999999999999999999999999999988817,

0.9999999999999999999999999999999999999893,

0.9999999999999999999999999999999999999921,

0.9999999999999999999999999999999999999921,

0.9999999999999999999999999999999999999921,

0.9999999999999999999999999999999999999921]

```
> exp_at_least_2_S_plus_H:=evalf(exp(Seval+Heval)-Seval-Heval-1)
  :
  poisson_at_least_2_S_plus_H:=[seq(0,i=1..21)]:
  poisson_at_least_2_S_plus_H[1]:=evalf((Seval+Heval)
  ^2/2/exp_at_least_2_S_plus_H):
  for i from 2 to 21 do
  poisson_at_least_2_S_plus_H[i]:=poisson_at_least_2_S_plus_H
  [i-1]+evalf((Seval+Heval)^(i+1)/(i+1)!
  /exp_at_least_2_S_plus_H);
  od:
  poisson_at_least_2_S_plus_H;
```

[0.9847226752646731123351474179133651081545, **(5.7.9)**

0.9998246825089710713085960972083725510078,

0.9999983892503733097821182943648635319675,

0.9999999876621086757235673281267270162434,

0.9999999999189735846072764943619199108758,

0.9999999999995343066240142607744805857090,

0.9999999999999976205491159308429811929007,

0.9999999999999999890569977358741268058598,

0.9999999999999999999542454852211293235191,

0.9999999999999999999998246255823708902305,

0.9999999999999999999999993794814164598366,

0.9999999999999999999999999979612059746853,

0.9999999999999999999999999999937476876956,

0.9999999999999999999999999999999820236046,

0.9999999999999999999999999999999999507940,

0.9999999999999999999999999999999999993123,

0.9999999999999999999999999999999999994363,

0.9999999999999999999999999999999999994366,

0.9999999999999999999999999999999999994366,

0.9999999999999999999999999999999999994366,

0.9999999999999999999999999999999999994366]

Calculating the choose-vectors of derivate networks

We recall the system verified by pointed networks

```
> equations_derivate_networks:={
  dD=dS+dP+dH,
  dS=(dP+dH)*x*D+(y+P+H)*D+(y+P+H)*x*dD,
  dP=y*(dS+dH)*exp(S+H)+(dS+dH)*(exp(S+H)-1),
  dH=dxK+dD*dyK
  }:
```

```
> solution_derivate_networks:=eval_derivate_networks(x0,y0);
```

solution_derivate_network := {*dH* = 0.3269127707345350747267155141772264693049, *dS* = 1.384891569022488102107252606323467168185, *dP* = 1.873000659606038667668790757894996357915, *dD* = 3.584804999363061844502758878395689995406} **(5.8.1)**

```
> dDeval:=subs(solution_derivate_networks,dD):dSeval:=subs
  (solution_derivate_networks,dS):dPeval:=subs
  (solution_derivate_networks,dP):dHeval:=subs
  (solution_derivate_networks,dH):dxKeval:=subs
  (solution_derivate_3_connected,dxK):dyKeval:=subs
  (solution_derivate_3_connected,dyK):
```

```
> choose_vector_dD_Bernoulli:=[evalf(dSeval/dDeval),evalf
  (dPeval/dDeval),evalf(dHeval/dDeval)];
  choose_vector_dD:=[choose_vector_dD_Bernoulli[1],
  choose_vector_dD_Bernoulli[1]+choose_vector_dD_Bernoulli[2]];
```

choose_vector_dD_Bernoulli := [0.3863227063309027336506058624601703661740, **(5.8.2)**

0.5224832759212364973024248816725136105229,

0.09119401774786076904696925586731602330273]

choose_vector_dD := [0.3863227063309027336506058624601703661740,

0.9088059822521392309530307441326839766969]

```
> choose_vector_dS_Bernoulli:=[evalf((dPeval+dHeval)*x0*
  Deval/dSeval),evalf((y0+Peval+Heval)*Deval/dSeval),evalf((y0+
  Peval+Heval)*x0*dDeval/dSeval)];
  choose_vector_dS:=[choose_vector_dS_Bernoulli[1],
  choose_vector_dS_Bernoulli[1]+choose_vector_dS_Bernoulli[2]];
```

choose_vector_dS_Bernoulli := [0.06637631584521710489998629353555705945149, **(5.8.3)**

0.8298003737791239976736787148530810837865,

0.1038233103756588974263349916113618567626]

choose_vector_dS := [0.06637631584521710489998629353555705945149,

0.8961766896243411025736650083886381432380]

```
> choose_vector_dP:=[evalf(y0*(dSeval+dHeval)*exp(Seval+Heval)
  /dPeval)];
```

choose_vector_dP := [0.9569684295031372171205429600262804987799] **(5.8.4)**

```
> choose_vector_dH:=[evalf(dxKeval/dHeval)];
```

choose_vector_dH := [0.7645864681449901009421614081667122271607] **(5.8.5)**

```
> ch_dP_or_dH:=[evalf(dPeval/(dPeval+dHeval))];
```

ch_dP_or_dH := [0.8513974385419680154225908152525064030393] **(5.8.6)**

```
> ch_dS_or_dH:=[evalf(dSeval/(dSeval+dHeval))];
```

ch_dS_or_dH := [0.8090244526538951837665827083281588238568] **(5.8.7)**

Calculating the choose-vectors of bi-derivate networks

We recall the system verified by bi-pointed networks

```
> equations_bi_derivate_networks:={
  ddD=ddS+ddP+ddH,
  ddS=(ddP+ddH)*x*D+2*(dP+dH)*D+2*(dP+dH)*x*dD+2*(y+P+H)*dD+(y+
  P+H)*x*ddD,
  ddP=y*(ddS+ddH)*exp(S+H)+y*(dS+dH)^2*exp(S+H)+(ddS+ddH)*(exp
  (S+H)-1)+(dS+dH)^2*exp(S+H),
  ddH=dxxK+2*dD*dxyK+ddD*dyK+dD^2*dyyK
  }:
> solution_bi_derivate_networks:=eval_bi_derivate_networks(x0,
  y0);solution_bi_derivate_3_connected:=subs({x=x0,op
  (eval_networks(x0,y0))},system_bi_derivate_3_connected);
```

solution_bi_derivate_network := {*ddD* **(5.9.1)**

= 3844.445282838549753803461158222513706993, *ddS*

= 314.6385745343491289148053195218860498557, *ddP*

= 2011.588639575041056089920779709191971176, *ddH*

= 1518.218068729159568798735058991435685964}

solutions_bi_derivate_3_connected ≔ {*dxyK*

= 72.098119886688268463880799694539689456 20, *dxxK*

= 840.103572875302590376914748297187630074 0, *dyyK*

= 6.121424587459627584125963350337009357995}

```
> ddDeval:=subs(solution_bi_derivate_networks,ddD):ddSeval:=subs
  (solution_bi_derivate_networks,ddS):ddPeval:=subs
  (solution_bi_derivate_networks,ddP):ddHeval:=subs
  (solution_bi_derivate_networks,ddH):dxxKeval:=subs
  (solution_bi_derivate_3_connected,dxxK):dxyKeval:=subs
  (solution_bi_derivate_3_connected,dxyK):dyyKeval:=subs
  (solution_bi_derivate_3_connected,dyyK):
```

```
> choose_vector_ddD_Bernoulli:=[evalf(ddSeval/ddDeval),evalf
  (ddPeval/ddDeval),evalf(ddHeval/ddDeval)];
  choose_vector_ddD:=[choose_vector_ddD_Bernoulli[1],
  choose_vector_ddD_Bernoulli[1]+choose_vector_ddD_Bernoulli[2]]
  ;
```

choose_vector_ddD_Bernoulli ≔ **(5.9.2)**

[0.08184238593247331757456964793851919317550,

0.5232454857804043737746168436168528655682,

0.3949121282871223086508135084446279412570]

choose_vector_ddD ≔ [0.08184238593247331757456964793851919317550,

0.6050878717128776913491864915553720587437]

```
> choose_vector_ddS_Bernoulli:=[evalf((ddPeval+ddHeval)*x0*
  Deval/ddSeval),evalf(2*(dPeval+dHeval)*Deval/ddSeval),evalf(2*
  (dPeval+dHeval)*x0*dDeval/ddSeval),evalf(2*(y0+Peval+Heval)*
  dDeval/ddSeval),evalf((y0+Peval+Heval)*x0*ddDeval/ddSeval)];
  choose_vector_ddS:=[choose_vector_ddS_Bernoulli[1],
  choose_vector_ddS_Bernoulli[1]+choose_vector_ddS_Bernoulli[2],
  choose_vector_ddS_Bernoulli[1]+choose_vector_ddS_Bernoulli[2]+
  choose_vector_ddS_Bernoulli[3],choose_vector_ddS_Bernoulli[1]+
  choose_vector_ddS_Bernoulli[2]+choose_vector_ddS_Bernoulli[3]+
  choose_vector_ddS_Bernoulli[4]];
```

choose_vector_ddS_Bernoulli ≔ [0.4687726682350615367469967761262876115633, **(5.9.3)**

0.01530056358901635049072095742828956577825,

0.001914382317267780736238243665484811007109,

0.02393256001326287716729915881967974222580,

0.4900798258453914548587448639602582694247]

choose_vector_ddS := [0.4687726682350615367469967761262876115633,

0.4840732318240778872377177335545771773416,

0.4859876141413456679739559772200619883487,

0.5099201741546085451412551360397417305745]

```
> choose_vector_ddP_Bernoulli:=[evalf(y0*(ddSeval+ddHeval)*exp
  (Seval+Heval)/ddPeval),evalf(y0*(dSeval+dHeval)^2*exp(Seval+
  Heval)/ddPeval),evalf((ddSeval+ddHeval)*(exp(Seval+Heval)-1)
  /ddPeval),evalf((dSeval+dHeval)^2*exp(Seval+Heval)/ddPeval)];

  choose_vector_ddP:=[choose_vector_ddP_Bernoulli[1],
  choose_vector_ddP_Bernoulli[1]+choose_vector_ddP_Bernoulli[2],
  choose_vector_ddP_Bernoulli[1]+choose_vector_ddP_Bernoulli[2]+
  choose_vector_ddP_Bernoulli[3]];
```

choose_vector_ddP_Bernoulli := [0.9540491337469277059706423164397971723511, **(5.9.4)**

0.001525283210087308768789387909129197470873,

0.04290029983289767649177890774194443270665,

0.001525283210087308768789387909129197470873]

choose_vector_ddP := [0.9540491337469277059706423164397971723511,

0.9555744169570150147394317043489263698220,

0.9984747167899126912312106120908708025286]

```
> choose_vector_ddH_Bernoulli:=[dxxKeval/ddHeval,2*dDeval*
  dxyKeval/ddHeval,ddDeval*dyKeval/ddHeval,dDeval^2*
  dyyKeval/ddHeval];
  choose_vector_ddH:=[choose_vector_ddH_Bernoulli[1],
  choose_vector_ddH_Bernoulli[1]+choose_vector_ddH_Bernoulli[2],
  choose_vector_ddH_Bernoulli[1]+choose_vector_ddH_Bernoulli[2]+
  choose_vector_ddH_Bernoulli[3]];
```

choose_vector_ddH_Bernoulli := [0.5533484221924193919260021788667853152309, **(5.9.5)**

0.3404750686847275323043754087552459005304,

0.05436223469557194484994328580548195892945,

0.05181427442728113091967912657248682530948]

choose_vector_ddH := [0.5533484221924193919260021788667853152309,

0.8938234908771469242303775876220312157613,

0.9481857255727188690803208734275131746908]

```
> ch_ddP_or_ddH:=[evalf(ddPeval/(ddPeval+ddHeval))];
```

ch_ddP_or_ddH := [0.5698863438733317955290547438303024329288] **(5.9.6)**

```
> ch_ddS_or_ddH:=[evalf(ddSeval/(ddSeval+ddHeval))];
```

ch_ddS_or_ddH := [0.1716656759222132629226192230232961943670] **(5.9.7)**

```
> solution_planar_graphs:=evaluate_planar_and_connected(z0,x0,y0);
```

solution_planar_graphs := {dC = 1.039822171978973903306270467977652135245, ddC = 1.18494484236796018449062644298827905779, dddC = 26.977173076155219138299636032417233275, G = 1.038147065671058036592476858173312080310, dG = 1.079488336659678024562533516788187660594, ddG = 2.352622917937788477130551013006437191000, dddG = 33.01085081533090095256556057640803402821, C = 0.03743745647180877655087152241832341434130} **(5.1)**

Calculating the choose-vectors of 2-connected planar graphs

```
> B_eval:=eval_2connected(x0,y0);dB_eval:=
  eval_derivate_2connected(x0,y0);ddB_eval:=
  eval_bi_derivate_2connected(x0,y0);dddB_eval:=
  eval_tri_derivate_2connected(x0,y0);
```

B_eval := 0.00073962188057625638648984729031898781819 **(5.10.1)**

dB_eval := 0.0390497100513283735279023911609909733436

ddB_eval := 1.051898476119285300579975636104216381688

dddB_eval := 17.98724517374004629815793198584565683

```
> ch_xy_in_dB:=[x0*y0/dB_eval];
```

ch_xy_in_dB := [0.9779613764891599617765672896377688574596] **(5.10.2)**

```
> ch_y_in_ddB:=[y0/ddB_eval];
```

ch_y_in_ddB := [0.9506620864108942651158377626844834749374] **(5.10.3)**

```
> ch_nontrivialD_or_dD:=[(D_eval-y0)/(x0*dDeval+D_eval-y0)];
```

ch_nontrivialD_or_dD := [0.4075315156808300986428709834840174147792] **(5.10.4)**

```
> ch_dD_or_ddD:=[dDeval/(dDeval+x0*ddDeval)];
```

ch_dD_or_ddD := [0.02383502205703488950551945928507080127172] **(5.10.5)**

Calculating the choose-vectors of connected planar graphs

```
> C_eval:=subs(solution_planar_graphs,C);dC_eval:=subs
  (solution_planar_graphs,dC);ddC_eval:=subs
  (solution_planar_graphs,ddC);dddC_eval:=subs
  (solution_planar_graphs,dddC);
```

C_eval := 0.03743745647180877655087152241832341434130 **(5.11.1)**

dC_eval := 1.039822171978973903306270467977652135245

ddC_eval := 1.18494484236796018449062644298827905779

dddC_eval := 26.977173076155219138299636032417233275

```
> exp_dB:=evalf(exp(dB_eval)):
  poisson_dB:=[seq(0,i=1..22)]:
  poisson_dB[1]:=evalf(1/exp_dB):
  for i from 2 to 22 do poisson_dB[i]:=poisson_dB[i-1]+evalf
  (dB_eval^(i-1)/(i-1)!/exp_dB):
  od:
  poisson_dB;
```

[0.9617029016575161644090151194677722784252, **(5.11.2)**

0.9992571211227633357086440505803855253763,

0.9999903618134242627695599818139397187029,

0.9999999060922136444235986553618247720552,

0.9999999992675434880228427269489253365908,

0.9999999999952374108887295211514190567229,

0.9999999999999734503377339534336327049414,

0.9999999999999998704759161357295207708913,

0.9999999999999999994382580770105886255184,

0.9999999999999999999978071948269541795638,

0.999999999999999999999922179079747449904,

0.999999999999999999999999746823174098327,

0.999999999999999999999999999239664943865,

0.999999999999999999999999999997879619226,

0.999999999999999999999999999999994480885,

0.999999999999999999999999999999999986534,

0.999999999999999999999999999999999999971,

1.000000000000000000000000000000000000000,

1.000000000000000000000000000000000000000,

1.000000000000000000000000000000000000000,

1.000000000000000000000000000000000000000,

1.000000000000000000000000000000000000000]

```
> ch_dC_or_ddC:=[evalf(dC_eval/(z0*ddC_eval+dC_eval))];
```

ch_dC_or_ddC := [0.9598289352871135523909375929792630048659] **(5.11.3)**

```
> ch_2ddC_or_dddC:=[evalf(2*ddC_eval/(z0*dddC_eval+2*ddC_eval))]
  ;
```

ch_2ddC_or_dddC := [0.7051839276326086024069249978467560740888] **(5.11.4)**

```
> choose_vector_dddC_Bernoulli:=[evalf((2*ddC_eval+z0*dddC_eval)
  *ddB_eval*dC_eval/dddC_eval),evalf((dC_eval+z0*ddC_eval)^2*
```

```
dddB_eval*dC_eval/dddC_eval),evalf((dC_eval+z0*ddC_eval)*
ddB_eval*ddC_eval/dddC_eval)];
choose_vector_dddC[1]+choose_vector_dddC[2]+choose_vector_dddC
[3];
choose_vector_dddC:=[choose_vector_dddC_Bernoulli[1],
choose_vector_dddC_Bernoulli[1]+choose_vector_dddC_Bernoulli
[2]];
```

$$choose\_vector\_dddC\_Bernoulli := [0.136258055370965020567158795853921027533 0, 0.813687717711331107985660727958834562785 0, 0.050054226917703871447180476187244410351 16] \quad (5.11.5)$$

$$choose\_vector\_dddC_1 + choose\_vector\_dddC_2 + choose\_vector\_dddC_3$$

$$choose\_vector\_dddC := [0.1362580553709650205671587958539210275330, 0.9499457730822961285528195238127555903180]$$

Calculating the choose-vectors of planar graphs

```
> G_eval:=subs(solution_planar_graphs,G);dG_eval:=subs
(solution_planar_graphs,dG);ddG_eval:=subs
(solution_planar_graphs,ddG);dddG_eval:=subs
(solution_planar_graphs,dddG);
```

$$G\_eval := 1.038147065671058036592476858173312080310 \quad (5.12.1)$$

$$dG\_eval := 1.079488336659678024562533516788187660594$$

$$ddG\_eval := 2.352622917937788477130551013006437191000$$

$$dddG\_eval := 33.01085081533090095256556057640803402821$$

```
> exp_C:=evalf(exp(C_eval)):
poisson_C:=[seq(0,i=1..21)]:
poisson_C[1]:=evalf(1/exp_C):
for i from 2 to 21 do
poisson_C[i]:=poisson_C[i-1]+evalf(C_eval^(i-1)/(i-1)!/exp_C)
od:
poisson_C;
```

$$[0.9632546611819397845416978125708145085422, \quad (5.12.2)$$

$$0.9993164656312055664193023893263942562695,$$

$$0.9999914967483877003086890305724743297400,$$

$$0.9999999205644099078376895278482274638362,$$

$$0.9999999994059713223176198158992703544048,$$

$$0.9999999999962968270424259729729011135994,$$

$$0.9999999999999802079403148857670610568700,$$

$$0.9999999999999999907427748286924292916656 5,$$

0.9999999999999999996150862000367702727941,

0.9999999999999999999985594723005564181651,

0.9999999999999999999999950986944041751288,

0.9999999999999999999999999847126432165110,

0.9999999999999999999999999999559844785590,

0.9999999999999999999999999999998823189423,

0.9999999999999999999999999999999997063343,

0.9999999999999999999999999999999999993133,

0.9999999999999999999999999999999999999988,

1.000000000000000000000000000000000000000,

1.000000000000000000000000000000000000000,

1.000000000000000000000000000000000000000,

1.000000000000000000000000000000000000000]

```
> choose_vector_ddG:=[evalf(ddC_eval*G_eval/ddG_eval)];
```

choose_vector_ddG := [0.5228832048293774359172171012578336714390] **(5.12.3)**

```
> choose_vector_dddG_Bernoulli:=[evalf(dddC_eval*
  G_eval/dddG_eval),evalf(2*ddC_eval*dG_eval/dddG_eval),evalf
  (dC_eval*ddG_eval/dddG_eval)];
  choose_vector_dddG:=[choose_vector_dddG_Bernoulli[1],
  choose_vector_dddG_Bernoulli[1]+choose_vector_dddG_Bernoulli
  [2]];
```

choose_vector_dddG_Bernoulli := **(5.12.4)**

[0.8483959782128407402217490283203170788872,

0.07749779877392303827421876514208462101516,

0.07410622301323622150403220653759830009757]

choose_vector_dddG := [0.8483959782128407402217490283203170788872,

0.9258937769867637784959677934624016999024]

All choose-vectors

```
> ch_1_or_u;
  ch_1_or_v;
  ch_u_or_v;
  ch_dxu_or_dxv;
  choose_vector_dxu;
  choose_vector_dxv;
  ch_dyu_or_dyv;
  choose_vector_dyv;
  choose_vector_dyu;
```

```
ch_K_in_dyK;
ch_dxK_in_dxyK;
ch_b_or_dxb;
ch_3b_or_dyb;
choose_vector_non_trivial_D;
choose_vector_D;
choose_vector_P;
ch_y_or_P_or_H;
ch_S_or_H;
poisson_S_plus_H;
poisson_at_least_1_S_plus_H;
poisson_at_least_2_S_plus_H;
choose_vector_dD;
choose_vector_dS;
choose_vector_dP;
choose_vector_dH;
ch_dP_or_dH;
ch_dS_or_dH;
choose_vector_ddD;
choose_vector_ddS;
choose_vector_ddP;
choose_vector_ddH;
ch_ddP_or_ddH;
ch_ddS_or_ddH;
ch_xy_in_dB;
ch_y_in_ddB;
ch_nontrivialD_or_dD;
ch_dD_or_ddD;
poisson_dB;
ch_dC_or_ddC;
ch_2ddC_or_dddC;
choose_vector_dddC;
poisson_C;
choose_vector_ddG;
choose_vector_dddG;
```

[0.6555410421528204868268866226466062451797] **(5.13.1)**

[0.2819959453606310344033925095368809490853]

[0.1710692050781203161735415238933961027360]

[0.2305097115910830985084671172699999783268]

[0.01070828643549451802688641337274819561550,
0.5053541432177472590134432066863740978076]

[0.5]

[0.2293834507206473691014551584061148930687]

[0.0063403237871936819234289354970883569921 76,
0.50317016189359684096171446774854417849 67]
[0.0043958336570008652822035772634551200093 65,
0.50219791682850043264110178863172756000 45]
[0.098872276218094921723042646700265140611 65]
[0.0034668460309401324302276685621587822827 99]
[0.014223801974055838758064441068052638026 67]
[0.017375618985605795878896897327161924056 96]
[0.46604506623203547879032731006401855924 06,
0.97745909721650850344853118555586144551 88]
[0.91393685900627443424108592005256099756 17,
0.95404616125083227324776492989891603684 02,
0.99806005910561851027223965363643806623 60]
[0.97768142778073375820477534290481381913 05]
[0.95212596719957411572036913959485694829 17,
0.99797899813081036693459256545621328532 90]
[0.95386502230321894509750733172780920682 68]
[0.95503344815752700539515023665497277895 42,
0.99897350076801020132916217465698846817 23,
0.99998431783789232819556142105349712060 75,
0.99999982003673010442627992500148666201 38,
0.99999999834656674527463642965621554423 93,
0.99999999998733516403125664179718846015 12,
0.99999999999991682644682848292020841538 66,
0.99999999999999952196610720790213346912 15,
0.99999999999999999755749549494556284700 09,
0.99999999999999999998876701658021216318 32,
0.99999999999999999999995303302571942335 78,
0.99999999999999999999999981997829499306 08,
0.99999999999999999999999999936303815056 08,
0.99999999999999999999999999997907179499 0,
0.99999999999999999999999999999993582006 4,

0.9999999999999999999999999999999999815474,

0.9999999999999999999999999999999999999496,

0.9999999999999999999999999999999999999994,

0.9999999999999999999999999999999999999995,

0.9999999999999999999999999999999999999995,

0.9999999999999999999999999999999999999995]

[0.9771719380310539506835922490163142274384,

0.9996512482842222455070277752132408328031,

0.9999959978414505514727594363181515406030,

0.9999999632297077054589908737677468542814,

0.9999999997183498522832957585512522730404,

0.9999999999981503239682933437721104538053,

0.9999999999999893691227544698529755475621,

0.9999999999999999456817477664058216948548,

0.9999999999999999997501924661882176435655,

0.9999999999999999999989555131012688823577,

0.9999999999999999999999959965419265948626,

0.9999999999999999999999999858347633220674,

0.9999999999999999999999999999534582836531,

0.9999999999999999999999999999998572718322,

0.9999999999999999999999999999999995896387,

0.9999999999999999999999999999999999988817,

0.9999999999999999999999999999999999999893,

0.9999999999999999999999999999999999999921,

0.9999999999999999999999999999999999999921,

0.9999999999999999999999999999999999999921,

0.9999999999999999999999999999999999999921]

[0.9847226752646731123351474179133651081545,

0.9998246825089710713085960972083725510078,

0.9999983892503733097821182943648635319675,

0.9999999876621086757235673281267270162434,

0.9999999999189735846072764943619199108758,

0.9999999999995343066240142607744805857090,

0.9999999999999976205491159308429811929007,

0.9999999999999999890569977358741268058598,

0.9999999999999999999542454852211293235191,

0.9999999999999999999998246255823708902305,

0.9999999999999999999999999379481416459836 6,

0.9999999999999999999999999979612059746853,

0.9999999999999999999999999999937476876956,

0.9999999999999999999999999999999820236046,

0.9999999999999999999999999999999999507940,

0.9999999999999999999999999999999999993123,

0.9999999999999999999999999999999999994363,

0.9999999999999999999999999999999999994366,

0.9999999999999999999999999999999999994366,

0.9999999999999999999999999999999999994366,

0.9999999999999999999999999999999999994366]

[0.3863227063309027336506058624601703661740,

0.9088059822521392309530307441326839766969]

[0.06637631584521710489998629353555705945149,

0.8961766896243411025736650083886381432380]

[0.9569684295031372171205429600262804987799]

[0.7645864681449901009421614081667122271607]

[0.8513974385419680154225908152525064030393]

[0.8090244526538951837665827083281588238568]

[0.08184238593247331757456964793851919317550,

0.6050878717128776913491864915553720587437]

[0.4687726682350615367469967761262876115633,

0.4840732318240778872377177335545771773416,

0.4859876141413456679739559772200619883487,

0.5099201741546085451412551360397417305745]

[0.9540491337469277059706423164397971723511,

0.9555744169570150147394317043489263698220,

0.9984747167899126912312106120908708025286]

[0.5533484221924193919260021788667853152309,

0.8938234908771469242303775876220312157613,

0.9481857255727188690803208734275131746908]

[0.5698863438733317955290547438303024329288]

[0.1716656759222132629226192230232961943670]

[0.9779613764891599617765672896377688574596,

0.0220386235108400382234327103622311425404 1]

[0.9506620864108942651158377626844834749374,

0.0493379135891057348841622373155165250626 0]

[0.4075315156808300986428709834840174147792,

0.5924684843191699013571290165159825852187]

[0.0238350220570348895055194592850708012717 2,

0.9761649779429651104944805407149291987283]

[0.9617029016575161644090151194677722784252,

0.9992571211227633357086440505803855253763,

0.9999036181342426276955998181393971870 29,

0.9999999060922136444235986553618247720552,

0.9999999992675434880228427269489253365908,

0.9999999999952374108887295211514190567229,

0.9999999999999734503377339534336327049414,

0.9999999999999998704759161357295207708913,

0.9999999999999999994382580770105886255184,

0.9999999999999999999978071948269541795638,

0.9999999999999999999999922179079747449904,

0.9999999999999999999999999746823174098327,

0.9999999999999999999999999999239664943865,

0.9999999999999999999999999999997879619226,

0.9999999999999999999999999999999994480885,

0.9999999999999999999999999999999999986534,

0.9999999999999999999999999999999999999971,

1.0000000000000000000000000000000000000000,

1.0000000000000000000000000000000000000000,
1.0000000000000000000000000000000000000000,
1.0000000000000000000000000000000000000000,
1.0000000000000000000000000000000000000000]

[0.9598289352871135523909375929792630048659]

[0.7051839276326086024069249978467560740888]

[0.1362580553709650205671587958539210275330,
0.9499457730822961285528195238127555903180]

[0.9632546611819397845416978125708145085422,
0.9993164656312055664193023893263942562695,
0.9999914967483877003086890305724743297400,
0.9999999205644099078376895278482274638362,
0.9999999994059713223176198158992703544048,
0.9999999999962968270424259729729011135994,
0.9999999999999802079403148857670610568700,
0.9999999999999999074277482869242929166565,
0.9999999999999999996150862000367702727941,
0.9999999999999999999985594723005564181651,
0.9999999999999999999999950986944041751288,
0.9999999999999999999999999847126432165110,
0.9999999999999999999999999999559844785590,
0.9999999999999999999999999999998823189423,
0.9999999999999999999999999999999997063343,
0.9999999999999999999999999999999999993133,
0.9999999999999999999999999999999999999988,
1.0000000000000000000000000000000000000000,
1.0000000000000000000000000000000000000000,
1.0000000000000000000000000000000000000000,
1.0000000000000000000000000000000000000000]

[0.5228832048293774359172171012578336714390]

[0.8483959782128407402217490283203170788872,
0.9258937769867637784959677934624016999024]